ÉCOLE DE TECHNOLOGIE SUPERIEURE
UNIVERSITE DU QUEBEC

MÉMOIRE PRESENTÉ A
L'ECOLE DE TECHNOLOGIE SUPERIEURE

COMME EXIGENCE PARTIELLE
A L'OBTENTION DE MAITRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE

PAR
ABDELKADER HASSANE

COMMANDE DE ROBOTS MANIPULANTS UN OBJET EN TEMPS RÉEL

MONTREAL, LE 11 AOUT 2003

CE MÉMOIRE A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE :

M. Maarouf Saad, directeur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Guy Gauthier, codirecteur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Jean-Pierre Kenné, président du jury
Département de génie mécanique à l'École de technologie supérieure

M. Charles Khairallah, membre du jury

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC
LE 25 AVRIL 2003
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMMANDE DE ROBOTS MANIPULANT UN OBJET EN TEMPS RÉEL

Abdelkader Hassane

Sommaire

Ce rapport présente un algorithme de commande position pour une structure robotique saisissant un objet. L'algorithme de commande adaptative de Slotine et Li est employé pour réaliser un contrôleur de position adaptatif à la variation des masses des liens pendant les mouvements sans contraintes. Cette loi de commande est composée d'une anticipation et d'un retour à un régulateur proportionnel dérivatif (PD). L'anticipation est construite à partir du modèle de la dynamique inverse du robot, et le contrôleur PD est implanté en conditionnant les vitesses et les accélérations articulaires.

De plus, un contrôleur de position proportionnel dérivatif fût implanté en temps réel sur la plate-forme QNX. Les paramètres du contrôleur ont été ajustés pour minimiser les erreurs de la poursuite de trajectoire.

L'implantation en temps réel est basée sur les macro-fonctions de la carte d'encodeurs S626 de Sensoray et de la carte PCIDCC5-P de ICS pour l'envoi du signal de commande PWM .

Aussi, un planificateur de trajectoires articulaires, à partir des trajets cartésiens rectilignes, a été développé à l'aide de la méthode de la déviation bornée (*Bounded Deviation Method*) de Taylor et en exploitant les outils les plus récents y compris les techniques des matrices creuses (*Banded Diagonal Matrix*).

Les résultats de simulation de la commande adaptative appliquée au robot *US Maker 100*, qui démontrent la stabilité et l'efficacité du contrôleur conçu, sont représentés dans un environnement graphique développé dans le cadre de ce projet. Cette interface permet à l'usager de définir les trajectoires désirées dans l'espace cartésien, modifier les paramètres de contrôle et visualiser le mouvement.

CONTROL OF ROBOTS HANDLING AN OBJET IN REAL TIME

Abdelkader Hassane

Abstract

This report presents an algorithm of position control for a 5-dof industrial robot grasping an object. This algorithm, developped by Slotine and Li, is employed to produce an adaptive position controller during the movements without constraints. This control law is made up from feed-forward and a feedback to a proportional derivative regulator (PD). The feed-forward is built starting from the model of the inverse dynamics of the robot. And the PD controller is established by sensing articular speeds and accelerations. Moreover, a position controller of proportional derivative was implemented in real time on QNX platform. The parameters of the controller were adjusted to minimize the tracking errors. The implantation in real time is based on the macro-functions of the S626 I/O card of Sensoray for position sensors and the PCIDCC5-P card of ICS for sending the PWM signals. Also, an articular trajectories generator, starting from the rectilinear cartesian ways, was developed using the Bounded Deviation Method (*BDM*) of Taylor and by exploiting the most recent tools including the techniques of the Banded Diagonal Matrix (*BDM*). The simulation results of the adaptive control applied to the US MAKER 100 robot, which show the stability and the good performance of the proposed controller, also illustrated in a graphical environment developed within the framework of this project. This interface helps the user to define the desired trajectories in cartesian space, and to modify the control parameters and to visualize the movement.

AVANT-PROPOS ET REMERCIEMENTS

Mon premier remerciement va à Dr. Maarouf Saad, mon directeur de mémoire qui non seulement a été à l'initiative de ce travail, mais de plus m'a apporté tout au long de ce travail ses conseils, son expérience et aussi sa convivialité.

Mon co-directeur de mémoire, M. Guy Gauthier, a très utilement complété l'encadrement de ce projet en apportant ses compétences.
Je voudrais également remercier Charles Khairallah, et Ernesto Corniels de leurs soutiens.

Plusieurs personnes ont suivi mon travail et ont apporté leur vision éclairée. Il s'agit de Zouhair Elguettioui, Omar Elouadghiri et Tarek Abi Aad membres du même groupe.

Enfin, je remercie ma famille de son soutien.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS ET DES SIGLES

a_i longueur du lien i du robot, m

α_i angle de torsion du lien i, radian

θ_i angle de rotation du lien i du robot, radian

d_i distance entre deux liens successifs i et i+1, m

T matrice de transformation homogène

R matrice de rotation

ψ ange de tangage (*pitch*), degré

γ angle de roulis (*roll*), degré

ω_i et $\dot{\omega}_i$ vitesse et accélération angulaire du lien i, rad/s et rad/s^2

v_i et $\dot{v}_i$ vitesse et accélération linéaire du lien i, m/s et m/s^2

$J(\theta)$ matrice jacobienne

m_i masse du lien i, Kg

${}^{j}v_{ci}$ vitesse linéaire du centre de masse du lien i dans le repère {j}, m/s

${}^{j}W_i$ vitesse angulaire du centre de masse du lien i dans le repère {j}, rad/s

${}^{j}P_{ci}$ vecteur du centre de masse du lien i dans le repère {j}, m

${}^{ci}I_j$ tenseur d'inertie du centre de masse du lien i dans le repère {j}, kg.m^2

τ_i couple appliqué au joint i, N.m

u_i énergie potentielle du lien i, J

k_i énergie cinétique du lien i, J

g accélération gravitationnelle, m/s^2.

M matrice de masse du robot

$q, \theta, \dot{q}, \dot{\theta}, \ddot{q}, \ddot{\theta}$ vecteur des positions, vitesses et accélérations articulaires

G matrice des termes de gravité

Vm matrice des forces centrifuges et de Coriolis

W matrice de régression dynamique

p vecteur des paramètres dynamiques

F vecteur des frottements visqueux

e vecteur d'erreur d'état

K_p et K_d matrices des gains proportionnels et dérivés du contrôleur PD

Λ matrice de pondération

(^) valeur estimée

(~) erreur d'estimation

$(_d)$ valeur désirée

V(t) fonction candidate de Lyapunov

ddl degré de libérté

DH convention de Denavitt-Hartenberg

DSP processeur de traitement numérique des signaux(*Digital Signal Processor*)

PC ordinateur personnel

A/D, D/A conversion analogique/digitale et digitale/analogique

PWM onde modulée en largeur d'impulsion(*Pulse Width Modulation*)

RAM mémoire vive(*Random Access Memory*)

INTRODUCTION

La robotique est définie comme étant l'étude consacrée aux machines qui peuvent remplacer l'homme dans l'exécution des tâches. Durant le XIX ème siècle, la révolution mécanique a abordé la substitution de l'homme dans la chaîne de fabrication. Ce concept a été introduit par le tchèque Karel Capek en 1921 [Sciavicco et Siciliano, 2000]. A cette occasion, le terme robot est inspiré du mot slave robota qui signifie main d'œuvre exécutive.

Et après le développement des sciences de fiction, le comportement conçu pour le robot a été conditionné pour le rendre de plus en plus similaire à son image virtuelle. Donc, un robot est une machine conçue pour exécuter une ou plusieurs tâches à plusieurs reprises, avec la vitesse et la précision souhaitée. Il y a autant de types de robots puisqu'il y a de types de tâches à exécuter. Un robot peut être commandé par un opérateur humain, parfois d'une grande distance. Mais la plupart des robots sont commandés par ordinateur, et tombent dans l'une ou l'autre de deux catégories: robots autonomes et robots d'insecte. La première catégorie permet une autonomie au robot à l'aide de simples commandes dictées par le concepteur. Quant à la deuxième catégorie, elle concerne les robots qui ont une similitude avec la forme d'insecte. Cette flotte est commandée par un contrôleur simple ; mais, dans l'ensemble, peut être sophistiqué.

Des robots sont parfois groupés selon la tranche de temps dans laquelle ils étaient des premiers largement répandus. Les robots de première génération datent des années 70 [Sciavicco et Siciliano, 2000] et se composent des dispositifs stationnaires, non programmables, électromécaniques sans sondes. Des robots de seconde génération ont été développés dans les années 80 et peuvent contenir des sondes et des contrôleurs programmables. Des robots de troisième génération ont été développés entre

approximativement 1990 et le présent [Sciavicco et Siciliano, 2000]. Ces machines peuvent être de type stationnaire ou mobile, autonome ou d'insecte, avec la programmation, la reconnaissance de la parole et/ou la synthèse, et autre dispositif sophistiqué plus avancé.

D'autre part, l'interprétation scientifique définit le robot comme étant une machine indépendante de son extérieur, capable de modifier son environnement où elle opère en accomplissant des actions conditionnées par certaines règles du comportement intrinsèque de la machine ainsi que les données nécessaires concernant l'environnement.

Autrement, la capacité du robot est fournie par le système mécanique qui est en général constitué par l'appareil de la locomotion pour le déplacement, et par l'appareil de manutention pour la manipulation d'objets; ainsi que la conception du système mécanique d'articulations, choix des matériaux, type des capteurs et des actionneurs qui assurent sa mobilité ; et en plus le système de contrôle qui garantit la supervision du mouvement du manipulateur.

De plus, le champ d'applications le plus connu des robots est le secteur industriel, qui présente trois utilisations fondamentales dans les procédés manufacturiers : les opérations de la manutention, les techniques de manipulation et les méthodes d'analyse et de la mesure. La première comprend à titre d'exemple la palettisation, la charge et la décharge, l'usinage mécanique et l'emballage, la deuxième englobe la soudure industrielle, la peinture, le perçage, et l'assemblage des circuits électroniques et la troisième concerne l'inspection, la détection et le contrôle de la qualité.

Cette évolution de la robotique est le fruit de plusieurs recherches faisant appel aux récentes théories développées dans les domaines de la mécanique, la télécommunication, l'électronique et la science de l'information. Ces recherches sont axées sur la modélisation de la structure mécanique du robot, de la planification de la trajectoire de

l'effecteur, des lois de contrôle et leur stabilité et l'architecture du matériel y compris les capteurs et les actionneurs.
Le but de ces recherches est la réalisation par le robot d'une tâche précise avec la meilleure performance. La tâche du manipulateur la plus étudiée est de suivre le mouvement ; cette tâche paraît facile à réaliser par une large gamme de robots ; mais quand la tâche devient plus complexe comme le transport d'un matériau flexible; et pour des considérations économiques, ergonomiques et de flexibilité, on fait recours à une coopération entre robots pour réaliser la tâche globale avec une haute performance.

Cet axe de recherche date juste des années 80 ; et le champ reste toujours ouvert puisque la majorité des travaux effectués jusqu'à ce jour ont étudié des tâches très spécifiques. La centralisation du contrôle de manipulateurs en coopération a été bien étudiée. Par contre, la décentralisation du contrôle et de la planification de la trajectoire pour des manipulateurs découplés faisait l'objet de récentes recherches en développant une nouvelle formulation d'échanges de données et du contrôle de la force durant la tâche. Parmi les études actuelles, on tire plusieurs approches et beaucoup de visions du problème. Le plus connu de ces concepts, est inspiré du module *maître – esclave*, dit aussi *chef-subordonnés,* étudiés particulièrement par [Sugar et Kumar , 2002].

Donc selon cette stratégie, la coopération entre plusieurs manipulateurs qui est la solution pratique pour réaliser des tâches complexes comprend trois niveaux :

* Le bas niveau : chaque manipulateur contrôle son propre mouvement pour éviter toute collision avec ses voisins ; se déplacer suivant une trajectoire désirée ; et maintenir l'objet stable. Cela fait appel à un contrôleur de la trajectoire combiné avec le contrôle de la force de contact au niveau de l'effecteur.

* Le niveau intermédiaire : il fallait établir une stratégie de coopération en manutention d'un objet ; dans le cas d'un système de deux robots. Elle pourrait être comme la structure chef-subordonnés où le chef planifie en poursuivant une trajectoire désirée, tandis que le subordonné maintient l'objet et garde sa position et son orientation relatives. Dans le cas d'un système de plus de deux robots, elle pourrait être plus qu'une structure chef-subordonné ou autre stratégie plus complexe. Donc il y a un nombre fini d'organisations de l'équipe de manipulateurs commandés séparément par des lois de commandes indépendantes.

* Le haut Niveau : il est nécessaire de planifier la trajectoire de toute l'équipe et concevoir des protocoles de coordination et d'échanges d'informations.

En s'inspirant de ces différentes approches, la présente étude contribuera à une coopération en temps réel de deux robots du laboratoire GREPCI de l'école pour réaliser une tâche classique. Il s'agit de transporter un objet suivant une trajectoire précise dans l'espace cartésien.

Pour être capable de contrôler les forces et les couples, qu'un robot applique sur un objet, il faut développer un modèle d'asservissement de force et créer le programme, qui permet d'interpréter les mesures et de commander en conséquence le robot.
Notre but est de développer un système de contrôle de mouvement qui soit le moins rigide et le plus flexible possible. Notre choix entre les architectures est, alors, déterminant. Notre système de contrôle de mouvement ne pourrait pas être adapté à d'autres robots utilisant un système d'exploitation différent. On est alors libre de choisir le langage de programmation pour remplir cette tâche. A ce niveau de l'étude, on peut remarquer que l'ordinateur occupe une place importante dans le système:

- Il peut remplacer le terminal.
- Il lit les données du couple provenant des cartes entrée/sortie.

- Il lit les données de mouvement et de position provenant du robot.
- Il détermine la trajectoire désirée des bras en exécutant les programmes de contrôle de mouvement.
- Il ordonne au contrôleur du robot de modifier les mouvements des bras.

L'objectif principal de ce travail se situe à la rencontre de ces deux aspects : la mobilité et les systèmes temps réel. Notre but est de proposer une plate-forme portable qui permet de piloter un robot ou tout autre contrôleur industriel qui doit respecter des contraintes temporelles. La première étape est de trouver un contrôleur numérique qui dispose de grande performance capable de réaliser des applications temps réel. La deuxième étape est de concevoir une interface graphique pour cette plate-forme matérielle. Il permettra de disposer de mécanismes simples pour communiquer avec l'extérieur. Il fournira le moyen d'exécuter plusieurs tâches simultanément en respectant les contraintes temps réel associées à chacune. La troisième étape est d'installer le système au complet. Pour ce faire, nous commençons de décrire brièvement les systèmes temps réel en rappelant la définition et les caractéristiques des systèmes temps réel suivi de la plate-forme temps réel adoptée pour ce projet au chapitre 1 ; ensuite nous abordons une modélisation fine de la structure mécanique du robot MAKER 100 de cinq degrés de liberté ; tout en déterminant les couples appliqués aux joints selon la méthode de Lagrange-Euler, accompagnée de la planification de la trajectoire basée sur le principe de Taylor au chapitre 2. Après, au chapitre 3, nous entamons la description de la commande adaptative tirée de la théorie de Slotine et Li avec une étude du cas du robot modélisé plus haut. Et au chapitre 4, nous proposons les techniques d'implantation en temps réel, suivi d'une discussion des résultats de la simulation. Et enfin, nous achevons par une conclusion et des recommandations.

CHAPITRE 1

REVUE DE LITTERATURE

L'application temps réel nous amenons à rappeler, au cours du présent chapitre, les définitions conventionnelles des systèmes embarqués, système d'exploitation et du temps réel. Ainsi une brève description de la plate-forme QNX est abordée, tout en mentionnant une comparaison entre un système classique et un autre temps réel.

1.1 Systèmes embarqués

Les systèmes embarqués sont apparus lors de la conquête spatiale quand il a fallu placer un système informatique à bord d'une navette. Le bon fonctionnement de ce système nécessitait la réalisation de calculs en un temps très court, ces calculs étant entre autres basés sur des informations collectées par des capteurs. De cette époque découle le terme anglo-saxon "embedded system" qui a été traduit par "embarqué" [Skivée, web, 2002].
De nos jours, de nombreux appareils utilisent de tels systèmes. Le terme "embarqué" pour un système dans un frigo n'était pas approprié, ce terme "embarqué" a été remplacé par "enfoui". Un système enfoui est un système informatique constituant un composant d'un système plus complexe. Aujourd'hui, une très grande part des systèmes informatiques fabriqués et utilisés dans le monde sont des systèmes enfouis. Ces systèmes sont évidemment présents dans les télécommunications et les transports mais aussi dans de nombreuses applications d'utilisation plus discrète, mais d'une importance tout aussi grande voire critique vis-à-vis du grand public. Ces systèmes sont de très grandes applications, d'une ingénierie complexe, faisant appel à de multiples compétences et soumis à des contraintes temps réel fortes.

L'aspect temps réel est fondamental pour les systèmes enfouis. Les systèmes temps réel doivent répondre dans un intervalle de temps donné et souvent court. Pour déterminer si un tel système fonctionne correctement, il faut non seulement vérifier qu'il produise des

résultats corrects mais qu'il les produise dans l'intervalle de temps fixé. Pour définir cet intervalle, on parle de contrainte temps réel. Le système temps réel garantit que quoi qu'il arrive, il fournira une réponse dans le délai défini par la contrainte temps réel.

Le système de gestion des sacs gonflables dans une voiture possède des contraintes temps réel très importantes. Quand un capteur détecte une déformation de la carrosserie suffisamment importante, il envoie un signal au contrôleur qui doit pouvoir gonfler les sacs gonflables dans les 10 ms sous peine d'arriver trop tard, ce qui aurait des conséquences désastreuses pour les occupants de la voiture. Même si un capteur tombe en panne et qu'il n'envoie plus d'informations, le système doit continuer à fonctionner et fournir les meilleures réponses possibles. Mais un système temps réel n'a jamais une seule tâche à effectuer, certains systèmes ont des dizaines de tâches à effectuer en même temps, chacune avec ses propres contraintes temps réel. Il faut donc un système principal qui gère l'ordonnancement de ces tâches. Dès que le système devient un peu complexe, c'est un système d'exploitation temps réel qui gère l'exécution des différentes tâches.

La différence principale entre un système d'exploitation généraliste et un système d'exploitation temps réel, c'est que le système temps réel est déterministe. Deux exécutions du même programme entraînent le même entrelacement des tâches et exactement la même exécution, ce qui n'est pas du tout garanti par un système d'exploitation généraliste. C'est cet aspect déterministe qui permet le respect de contraintes temps réel.

1.2 Principe fondamental du système d'exploitation

Il est possible de résumer le travail d'un système d'exploitation à deux tâches essentielles [Mabilleau, web, 2002] :

1) Pour le programmeur, le système d'exploitation offre une interface de programmation commode. En effet, l'architecture d'un ordinateur est primaire et

lourde à programmer en langage machine. Le système d'exploitation est le logiciel qui soustrait le matériel aux regards du programmeur et offre une vue simple et facile à programmer de la machine (on parle alors de machine virtuelle).

2) Parallèlement, le système d'exploitation contrôle l'accès à l'ensemble des ressources d'une machine : mémoire vive, mémoires de masse, périphériques d'affichages, etc. En effet, plusieurs programmes peuvent s'exécuter en même temps (multiprogrammation) et plusieurs utilisateurs peuvent utiliser simultanément une même machine (cas des systèmes Multi-utilisateurs). Le système d'exploitation est en charge d'allouer, avec un ordre de priorité, les ressources aux différents demandeurs.

1.3 Définition du Temps Réel

Luc Aleaume et Laurent Grzybek [Phillipe, web, 2001] nous donnent une définition de la qualité temps réel comme étant *l'aptitude d'une application à répondre à des sollicitations de l'environnement contrôlé, en des temps de réponse en relation avec la constante de temps dominante dans cet environnement.* On voit donc qu'un système temps réel consiste en une partie contrôlante agissant sur une partie contrôlée avec certaines contraintes de temps. Une définition plus précise d'une application temps réel peut s'écrire comme étant la capacité d'une application à pouvoir appréhender un flux d'évènements et de traiter chacun d'eux en un temps déterminé. Le concept clef du temps réel est le déterminisme. Dire d'un système qu'il est prévisible (déterministe) ou qu'il est temps réel est quasiment équivalent. Citons, comme exemple, que le contrôle d'un robot est temps réel s'il est possible d'exécuter un programme de contournement d'objet dans les 500ms qui suivent la détection d'un obstacle. Le système de contrôle d'une centrale nucléaire est temps réel si le délai de 2ms de prise en compte et de traitement (mise en route du refroidissement, arrêt de la fission) d'une surchauffe est respecté. Un système

temps réel n'est donc pas nécessairement un système rapide [Salvetti, web, 2002], mais il est capable de donner l'illusion de la réalité (simulations, jeux interactifs, etc.).

1.4 Comparaison entre un système temps réel et un système classique

Pour mettre en évidence la différence entre un système classique et un système temps réel, un test simple peut être mis en œuvre. On génère un signal périodique sur le port parallèle de la machine [Blachier, web, 2001]. En mesurant, à l'aide d'un compteur branché en sortie du port parallèle, le temps qui sépare deux fronts du signal, on peut constater les différences entre un système classique et un système temps réel.

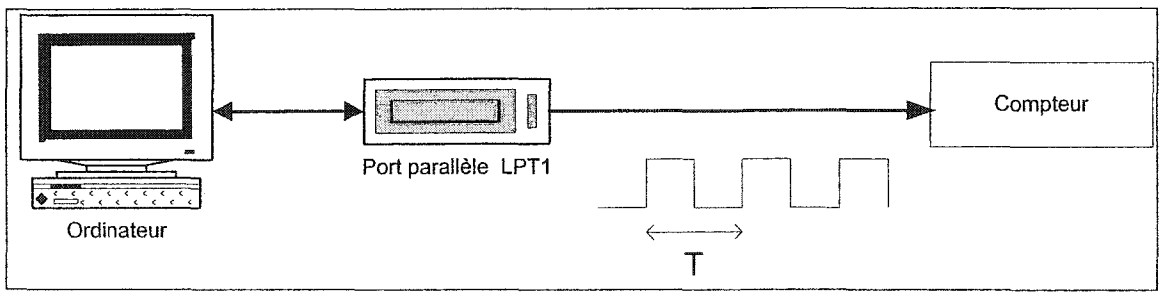


Figure 1 Montage électronique [Blachier, 2001]

Le montage utilisé est celui de la figure 1. On souhaite générer un signal carré de fréquence 25 Hz (i.e. de demi-période T/2 = 20 ms). Avec un système classique, la demi-période du signal varie de 17 ms à 23 ms. La fréquence correspondante est donc comprise entre 22 Hz et 29 Hz. Avec un système temps réel, la dispersion de la demi-période est beaucoup plus faible. Elle varie de 19,990 ms et 20,015 ms. La fréquence varie alors de 24,98 Hz à 25,01 Hz. On constate donc l'importance des systèmes temps réel. Si en l'absence totale de charge le signal émit a une fréquence correcte quel que soit le système, ce n'est plus le cas lorsque la charge est maximale. Le système temps réel parvient à assurer une fréquence variant de +/- 0,2 Hz contre +/- 4 Hz avec un système classique [Blachier, web, 2001].

1.5 Choix du système d'exploitation

Lors du choix du système d'exploitation, il faut considérer tout un ensemble de contraintes :

- La taille en mémoire :

  la carte mère ne disposant que d'un espace mémoire réduit , notre système doit prendre le moins de place possible en mémoire afin de pouvoir non seulement s'installer intégralement sur le disque mais également laisser le plus de place possible pour les fichier exécutables des autres programmes commandant le robot.

- Le coût.
- Le temps réel qui n'est pas indispensable mais est un critère de choix important dans un système embarqué.
- Les outils de développement.
- La documentation.
- La disponibilité du code source.
- Le support du matériel :

  S'il existe déjà des pilotes de périphériques pour le matériel que nous avons l'intention d'implanter sur le robot, il ne sera pas nécessaire de les programmer. L'écriture de certains pilotes étant même inenvisageable par manque de compétence ou du temps, il est indispensable que certains matériels soient déjà supportés dans le noyau du système d'exploitation. C'est donc un critère de choix de plus.

1.6 Réponse déterministe

La première apparition du terme temps réel se situe en 1970 et coïncide avec l'apparition des microprocesseurs dans l'environnement industriel. En informatique industrielle, le temps réel signifie le traitement déterministe d'événements survenant sur des procédés généralement rapides. Le temps réel est la capacité d'un système à répondre de manière fiable et déterministe (dans un laps de temps garanti). Ce délai minimum exigé dépend des applications et il ne faut donc pas confondre temps réel avec rapidité. Un exemple

classique est la régulation : si un régulateur n'était pas déterministe (temps de réponse variable), le système régulé pourrait devenir instable. C'est le degré d'incertitude sur ces délais qui nous indique si un système est temps réel ou non.

Donc, les systèmes temps réel (hard real time) ne peuvent en aucun cas tolérer une réponse tardive, les conséquences d'une telle réponse peuvent être catastrophiques (ex : système de pilotage automatique d'avion). «*Un résultat juste, mais hors délai, est un résultat faux*» [Abrial, web, 2002]. Pour commander un environnement industriel, il faudra respecter les contraintes de temps. Le temps d'exécution d'une tâche doit être connu et non soumis à des variations liées à la charge du système. Au niveau logiciel, on distingue : l'exécutif Temps Réel (le noyau) et l'application Temps Réel (les tâches). Pour fournir une réponse, un système doit reconnaître, traiter et sortir un résultat.
Le temps de réponse TR est la somme du temps de calculs et celui des opérations d'entrée/sortie. L'apparition d'un phénomène implique l'exécution d'une action effective au plus tard dans un délai TR appelé temps de réponse. Un système sera dit "déterministe" lorsque le temps maximal qu'il mettra pour traiter une tâche quelconque sera connu et déterminé à l'avance.

1.7 La plate-forme QNX

Depuis 1990, l'éditeur canadien QNX Software Systems [Ripoll, web, 2001] développe l'OS temps réel QNX, comme le premier OS temps réel à micro-noyau certifié POSIX. En 1996, QNX/Neutrino est le premier système POSIX embarqué. Et en 1999, QNX annonce le support multi-plate-formes : x86, Mips et PowerPC.

QNX se destine avant tout aux systèmes embarqués. Ce marché en pleine effervescence recouvre tout équipement autre qu'un PC : téléphone, ordinateurs de poche, automobile, électroménager, etc. Mais QNX ne se limite pas à ces équipements ; il tourne également sur les systèmes à base d'Intel x86, de Mips et de PowerPC. Extensible, il peut se loger dans un système autonome avec 32 Ko de mémoire, ou gérer un super ordinateur de

plusieurs centaines de processeurs. À titre d'exemple, sur PC, le système avec une interface graphique, un navigateur, un serveur Web et les services TCP/IP nécessaires peuvent tenir sur une disquette 1,44 Mo.

De plus, pour surmonter les contraintes du temps réel, un système temps réel doit offrir des temps de réponse déterministes et une fiabilité à toute épreuve. QNX s'appuie sur son architecture micro-noyau pour satisfaire à ces exigences. Contrairement aux OS à noyau monolithique comme Linux (ce qui n'est plus le cas avec la toute dernière version bêta), le noyau du système n'implémente que les services fondamentaux (communication inter-tâches, ordonnancement, gestion des interruptions). Les services de haut niveau (fichier, entrées/sorties, réseau) et les pilotes sont implémentés dans des processus séparés. Cette architecture offre une protection mémoire complète (aucun processus ne peut bloquer le système, pas même un pilote) et des temps de commutation entre tâches très rapides. Il permet également une grande extensibilité de l'OS. La conformité POSIX permet à QNX d'être portable sur le terrain des OS embarqués, il entre en compétition avec Windows CE et Windows NT Embedded, mais ces OS sont bien plus gourmands en ressources que lui. Quant à JavaOS, IBM et Sun ont récemment annoncé son abandon. On a ainsi pu voir une démonstration de robots contrôlés par des programmes temps réel écrits en Java. La fiabilité et la portabilité matérielle de QNX, combinées à la portabilité logicielle de Java est donc garantie. D'autre part, parmi les qualités de QNX : excellente fiabilité ; pas d'applications standard ; temps de réponse ; support de nombreux processeurs ; extensible. Le but de QNX est de servir de base pour un développement spécifique sur un matériel donné. Nous pouvons citer les domaines où QNX a trouvé une utilisation : automatisation industrielle, instrumentation, finance et points de vente, télécoms, informatique de poche, électronique grand public.

CHAPITRE 2

MODÉLISATION DU ROBOT MAKER 100

Dans ce chapitre, nous donnons l'étude du robot sur lequel nous effectuerons nos tests, en commençant par l'étude de la cinématique directe et inverse, après nous passons à la modélisation dynamique du robot.

2.1 Cinématique directe et inverse

2.1.1 Systèmes d'axes

La figure 2 présentée ci-dessous montre la configuration du robot MAKER 100 de « United States Robots » qui possède cinq degré de liberté. Il s'agit d'axes rotatifs pour les axes 1, 2, 4 et 5 et d'un axe prismatique pour l'axe 3. A signaler que les deux derniers contrôlent l'angle de tangage (pitch) et l'angle de roulis (roll) respectivement.

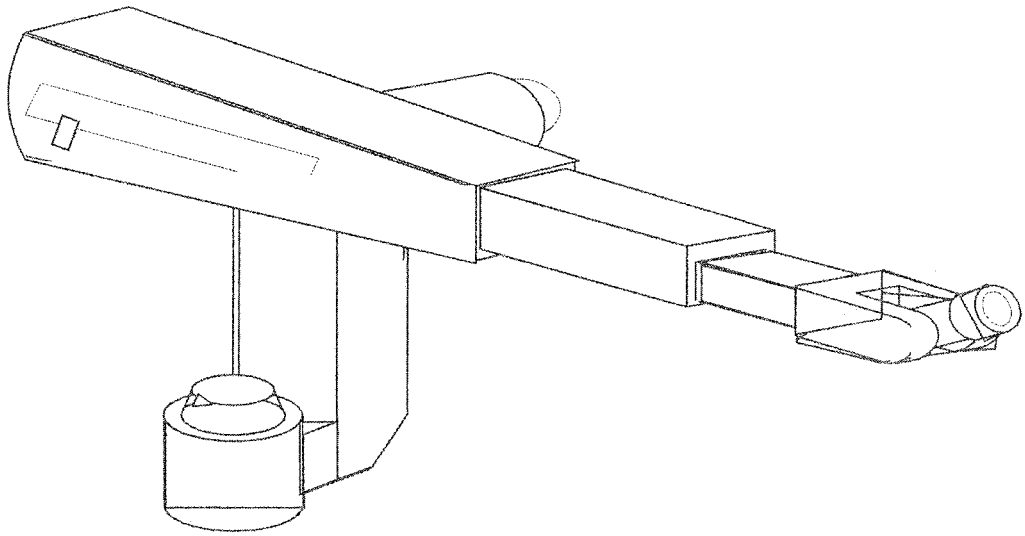


Figure 2 Le robot MAKER 100 à cinq degrés de liberté

La figure 3 illustre les systèmes d'axes choisi pour chaque articulation du robot MAKER 100.

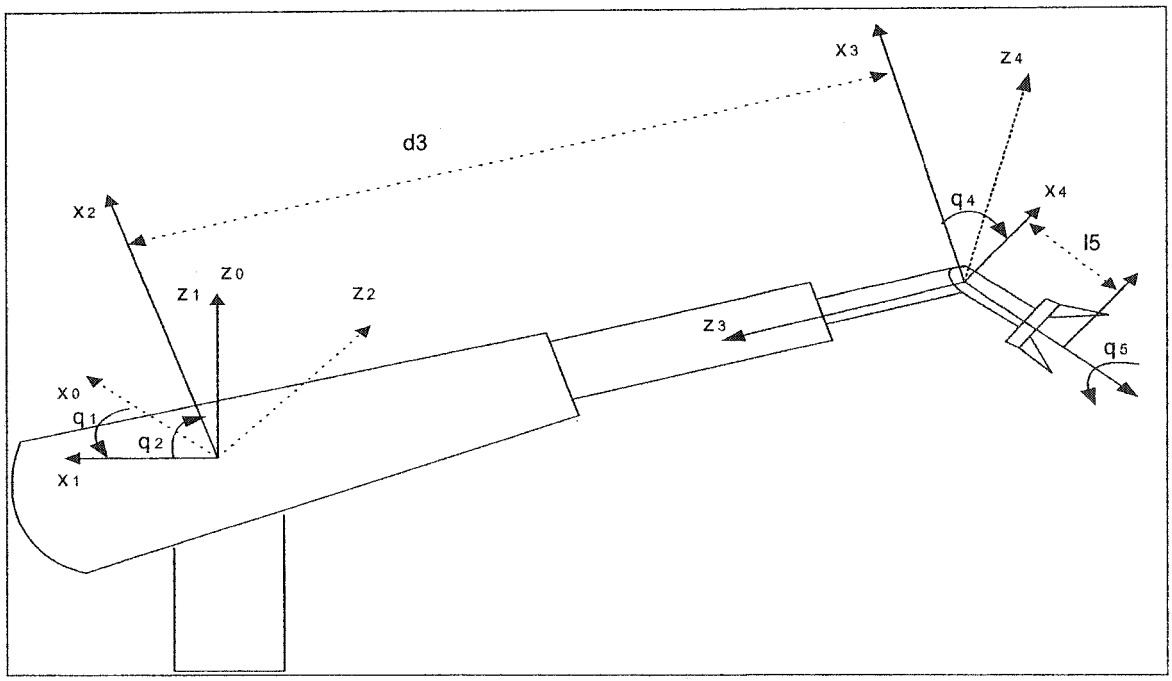


Figure 3 Les systèmes d'axes du robot MAKER 100

Lors de l'établissement des systèmes d'axes (fig. 3), on a suivi les conventions de [Craig, 1986] pour déterminer les paramètres de Denavit-Hartenberg modifiés (DH) du robot comme suit :

a) On choisit un système de coordonnée fixe qui sera considéré comme référence pour tous les mouvements.

b) On fixe les axes OZ des rotations (respectivement de translation) pour les articulations angulaires (respectivement prismatique). Le sens positif peut être décidé d'une façon arbitraire (voir figure 4).

Puisque la dernière articulation est rotative, on choisit son origine (O_5) de telle sorte que d_5 soit égale à 0, donc confondu avec l'origine O_4.

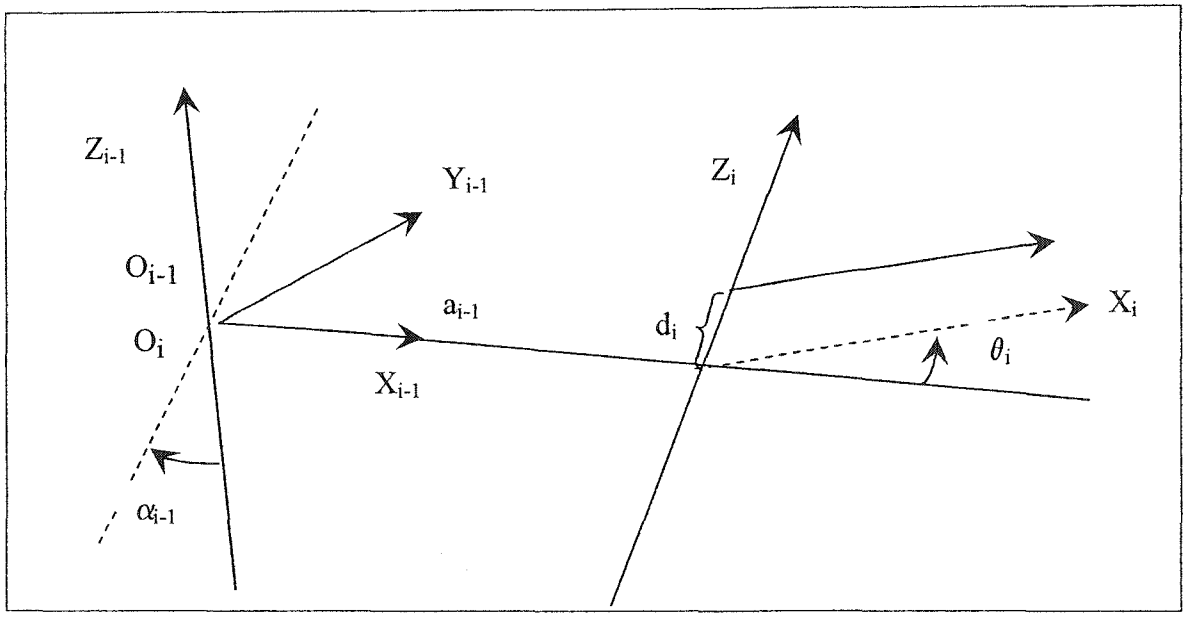


Figure 4 Convention de système d'axes

On définit ces paramètres par :

- a_{i-1} : la distance perpendiculaire commune entre les axes *i-1* et *i* (attachée au membre *i-1*), i.e. : $a_{i-1} = \|\overrightarrow{O_{i-1}O_i}\|$.
- α_{i-1} : l'angle de torsion (fixe) : c'est l'angle nécessaire pour ramener l'axe *i* parallèle à l'axe *i-1*, i.e. : $\alpha_{i-1} = \langle Zi-1, Zi \rangle / Xi-1$.
- θ_i : l'angle de la rotation autour de Zi de Xi-1 vers Xi., i.e. : $\theta_i = \langle Xi-1, Xi \rangle / Zi$.
- d_i : translation selon Zi des points de rencontre avec Xi-1 et Xi., i.e. : $d_i = \overrightarrow{O_{i-1}O_i}.Zi$.

2.1.2 Paramètres DH du robot

Le tableau I montre les différents paramètres attachés au robot MAKER 100, selon la convention de Denavitt-Hartenberg.

Tableau I

Paramètres des différentes articulations

articulation	variable	α_{i-1}	a_{i-1}	d_i	θ_i
1	θ_1	0	0	0	θ_1
2	θ_2	90	0	0	θ_2
3	d_3	90	0	$-d_3$	0
4	θ_4	-90	0	0	θ_4
5	θ_5	-90	0	0	θ_5

2.1.3 Espace de travail du robot

Le tableau II résume les différentes limites de déplacement relatives à chaque articulation du robot MAKER 100.

Tableau II

L'espace de travail du robot

Articulation	Type	Marge de manœuvre
# 1	Rotative	entre 0° et +350°
# 2	Rotative	entre -141° et +141°
# 3	Prismatique	entre 400 mm et 910 mm
# 4	Rotative	entre -114° et + 114°
# 5	Rotative	entre 0° et +348°

2.1.4 Matrice Globale de transformation homogène

À l'aide des outils de calcul symbolique, la matrice globale de transformation homogène du robot, calculée en détail dans l'annexe 1, est donnée par :

$$ {}^{0}_{5}T = \begin{bmatrix} c_1.c_5.c_{24} - s_1.s_5 & -c_1.s_5.c_{24} - s_1.c_5 & -c_1.s_{24} & -c_1.s_2.d_3 \\ s_1.c_5.c_{24} + c_1.s_5 & -s_1.s_5.c_{24} + c_1.c_5 & -s_1.s_{24} & -s_1.s_2.d_3 \\ c_5.s_{24} & -s_5.s_{24} & c_{24} & c_2.d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1) $$

De plus, le point du contact Pc de l'objet a les coordonnées suivantes par rapport au repère {5} : Pc=[0 0 l_5]^T; où l_5 est la distance du point Pc à l'origine du repère {5}.
Ce qui nous donne l'expression résultante suivante :

$$ {}^{0}_{c}T = {}^{0}_{5}T.{}^{5}_{c}T = \begin{bmatrix} c_1.c_5.c_{24} - s_1.s_5 & -c_1.s_5.c_{24} - s_1.c_5 & -c_1.s_{24} & -c_1.(s_2.d_3 + s_{24}.l_5) \\ s_1.c_5.c_{24} + c_1.s_5 & -s_1.s_5.c_{24} + c_1.c_5 & -s_1.s_{24} & -s_1.(s_2.d_3 + s_{24}.l_5) \\ c_5.s_{24} & -s_5.s_{24} & c_{24} & c_2.d_3 + c_{24}.l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2) $$

2.1.5 Étude de la cinématique inverse

Afin d'étudier la cinématique inverse, nous assumons qu'on a un point dans l'espace définit par une matrice 4 par 4, déterminé en étudiant la cinématique directe. Cette matrice définit les coordonnées (position et orientation) de ce point dans l'espace.

Donc, à partir des systèmes 2.3 et 2.4, on cherche à déterminer le vecteur d'articulations.

$$ \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} -c_1.(s_2.d_3 + s_{24}.l_5) \\ -s_1.(s_2.d_3 + s_{24}.l_5) \\ c_2.d_3 + c_{24}.l_5 \end{bmatrix} \quad (2.3) $$

D'autre part, une comparaison entre la matrice ${}^{0}_{c}T$ et celle des angles d'Euler est représentée par :

$$ Rot(z,\varphi).Rot(y,\phi).Rot(x,\gamma) = \begin{bmatrix} c_\varphi.c_\phi.c_\gamma - s_\varphi.s_\gamma & -c_\varphi.c_\phi.s_\gamma - s_\varphi.s_\gamma & -c_\varphi.s_\phi \\ s_\varphi.c_\phi.c_\gamma + c_\varphi.s_\gamma & -c_\varphi.c_\phi.s_\gamma + c_\varphi.c_\gamma & -s_\varphi.s_\phi \\ s_\phi.c_\gamma & -s_\phi.s_\gamma & c_\phi \end{bmatrix} \quad (2.4) $$

Et en identifiant les éléments des deux matrices, on peut déduire les trois angles d'Euler :

$$ \varphi = \theta_1 \qquad \psi = \theta_2 + \theta_4 \quad \gamma = \theta_5 \quad (2.5) $$

En combinant les systèmes 2.5 et 2.3, on aboutit au système 2.6 :

$$ \begin{bmatrix} p_x \\ p_y \\ p_z \\ \psi \\ \gamma \end{bmatrix} = \begin{bmatrix} -c_1.(s_2.d_3 + s_{24}.l_5) \\ -s_1.(s_2.d_3 + s_{24}.l_5) \\ c_2.d_3 + c_{24}.l_5 \\ \theta_2 + \theta_4 \\ \theta_5 \end{bmatrix} \quad (2.6) $$

Et par la suite, on fait la cinématique inverse, et on aura :

$$\begin{bmatrix} d_3 \\ \theta_2 \\ \theta_1 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \begin{bmatrix} \left[(p_z - c_\psi . l_5)^2 + (\varepsilon\sqrt{p_x{}^2 + p_y{}^2} - s_\psi . l_5)^2\right]^{1/2} \\ A\tan 2(\varepsilon\sqrt{p_x{}^2 + p_y{}^2} - s_\psi . l_5, p_z - c_\psi . l_5) \\ A\tan 2(\frac{p_y}{Ac}, \frac{p_x}{Ac}) \\ \psi - \theta_2 \\ \gamma \end{bmatrix} \tag{2.7}$$

avec :

$\varepsilon = \mp 1.$

$Ac = -(s_2 . d_3 + s_\psi . l_5)$

2.1.6 Matrice Jacobienne

Pour déterminer la matrice jacobienne, on doit exprimer les vitesses angulaires et linéaires du poignet qui sont données par les équations suivantes (voir annexe 1 pour plus de détails sur le calcul de ces vitesses) :

$${}^0w_5 = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} s_1 \dot{\theta}_2 - c_1 s_{24} \dot{\theta}_5 + s_1 \dot{\theta}_4 \\ -c_1 \dot{\theta}_2 - s_1 s_{24} \dot{\theta}_5 - c_1 \dot{\theta}_4 \\ c_{24} \dot{\theta}_5 + \dot{\theta}_1 \end{bmatrix} \tag{2.8}$$

$${}^0v_5 = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} s_1 . s_2 . d_3 . \dot{\theta}_1 - c_1 . c_2 . d_3 . \dot{\theta}_2 - c_1 . s_2 . \dot{d}_3 \\ -c_1 . s_2 . d_3 . \dot{\theta}_1 - s_1 . c_2 . d_3 . \dot{\theta}_2 - s_1 . s_2 . \dot{d}_3 \\ -s_2 . d_3 . \dot{\theta}_2 + c_2 . \dot{d}_3 \end{bmatrix} \tag{2.9}$$

Par définition, la matrice Jacobienne lie la vitesse cartésienne et la vitesse articulaire comme suit :

$$\begin{bmatrix} V_x \\ V_y \\ V_z \\ W_x \\ W_y \\ W_z \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \end{bmatrix} \tag{2.10}$$

Avec :

$$J = \begin{bmatrix} s_1 s_2 d_3 & -c_1 c_2 d_3 & -c_1 s_2 & 0 & 0 \\ -c_1 s_2 d_3 & -s_1 c_2 d_3 & -s_1 s_2 & 0 & 0 \\ 0 & -s_2 d_3 & c_2 & 0 & 0 \\ 0 & s_1 & 0 & s_1 & -c_1 s_{24} \\ 0 & -c_1 & 0 & -c_1 & -s_1 s_{24} \\ 1 & 0 & 0 & 0 & c_{24} \end{bmatrix} \tag{2.11}$$

2.2 Modélisation dynamique du robot par la méthode de Lagrange

Nous avons formulé le modèle dynamique du robot, en appliquant la méthode de Lagrange choisie arbitrairement, tout en supposant que la structure mécanique est rigide. Cette méthode consiste à établir un bilan énergétique du système mécanique.

Le lagrangien $L(\theta, \dot{\theta})$ est définit comme étant la différence entre l'énergie cinétique $k(\theta, \dot{\theta})$ et l'énergie potentielle *u(q)* :

$$L(\theta, \dot{\theta}) = k(\theta, \dot{\theta}) - u(\theta). \tag{2.12}$$

De plus, les équations de mouvement de Lagrange pour un manipulateur sont données par :

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{\theta}}\right]-\left[\frac{\partial L}{\partial \theta}\right]=\tau \tag{2.13}$$

Soit encore :

$$\frac{d}{dt}\left[\frac{\partial k}{\partial \dot{\theta}}\right]-\left[\frac{\partial k}{\partial \theta}\right]+\left[\frac{\partial u}{\partial \theta}\right]=\tau \tag{2.14}$$

Ou de manière équivalente :

$$\frac{d}{dt}\left[\frac{\partial k}{\partial \dot{\theta}_i}\right]-\left[\frac{\partial k}{\partial \theta_i}\right]+\left[\frac{\partial u}{\partial \theta_i}\right]=\tau_i \quad , \quad i=1...5 \tag{2.15}$$

Les τ_i représentent les couples exercés extérieurement par les actionneurs dans l'articulation i.

Pour ce faire, nous calculons les énergies cinétiques et potentielles du système mécanique puis nous procéderons à une dérivation pour aboutir au modèle dynamique du robot.

2.2.1 Énergie Potentielle

L'énergie potentielle totale u est la somme des énergies potentielles u_i des centres de masse des articulations i=1,..,5. Soit alors :

$$u(\theta)=-\sum_{i=1}^{5} m_i g_i^T \cdot {}^i p_{ci} \tag{2.16}$$

où :

m_i est la masse de l'articulation i

g_i est le vecteur de gravité de l'articulation i.

${}^i p_{ci}$ est le vecteur de coordonnées du centre de masse de l'articulation i dans le référentiel $\{i\}$.

2.2.2 Vecteurs de gravité et des centres de masse

Les coordonnées des centres de masse des articulations i exprimées dans les repères relatifs $\{i\}$ et le vecteur de gravité calculé dans la base $\{0\}$, sont regroupées dans le tableau III suivant :

Tableau III

Coordonnées des centres de masse et du vecteur de gravité

Articulation	Centre de masse	Vecteur de gravité
1	$[0 \quad 0 \quad z_1]^T$	$[0 \quad 0 \quad -g]^T$
2	$[0 \quad y_2 \quad 0]^T$	$[0 \quad 0 \quad -g]^T$
3	$[0 \quad 0 \quad z_3]^T$	$[0 \quad 0 \quad -g]^T$
4	$[0 \quad y_4 \quad 0]^T$	$[0 \quad 0 \quad -g]^T$
5	$[0 \quad 0 \quad z_5]^T$	$[0 \quad 0 \quad -g]^T$

Nous trouvons alors :

$$u = g(p_0 + (p_{12} - \frac{1}{2}.p_5).c_2 + p_6.c_{24} + p_4.c_2.d_3) \tag{2.17}$$

où :

$p_0 = m_1.z_1$

$p_4 = m_4 + m_5 + m_6$

$p_5 = 2.m_3.z_3$

$p_6 = m_5.z_5$

$p_{12} = m_2.y_2$

2.2.3 Énergie cinétique

L'énergie cinétique totale est la somme des énergies cinétiques de différentes articulations i :

$$k(\theta,\dot{\theta}) = \sum_{i=1}^{i=5} k_i \tag{2.18}$$

Avec :

$$k_i = \frac{1}{2}.m_i.{}^0v_{ci}^T.{}^0v_{ci} + \frac{1}{2}.{}^iw_i^T.{}^{ci}I_i.{}^iw_i \qquad \text{pour } i=1 \text{ à } 5. \tag{2.19}$$

où :

m_i est la masse de l'articulation i.

${}^iw_i$ est la vitesse angulaire du centre de masse de l'articulation i dans le repère i.

${}^0v_{ci}$ est la vitesse linéaire du centre de masse de l'articulation i dans le repère 0. Elle est calculée par l'équation suivante :

$${}^0v_{ci} = \frac{d}{dt}\left[{}^0P_{ci}\right]. \tag{2.20}$$

${}^{ci}I_i$ est le tenseur d'inertie au centre de masse de l'articulation i dans le repère i. Il est, généralement, sous la forme diagonale suivante :

$${}^{ci}I_i = \begin{pmatrix} I_{xxi} & 0 & 0 \\ 0 & I_{yyi} & 0 \\ 0 & 0 & I_{zzi} \end{pmatrix} \qquad i=1 \text{ à } 5 \tag{2.21}$$

L'énergie cinétique k_i est en effet composée de deux termes, le premier représente l'énergie cinétique due à la vitesse linéaire du centre de masse de l'articulation i ; et le second représente celle due à la vitesse angulaire de cette articulation.

À l'aide des outils informatiques et après simplification des formules trouvées ; l'énergie cinétique totale s'écrit :

$$k = \frac{1}{2}\left(- p_5.d_3.s_2^2 + p_4.d_3^2.s_2^2 + 2.p_6.s_2.s_{24}.d_3 + p_2.c_2^2 + p_7.c_{24}^2 + p_1 + p_3.s_2^2 + p_8.s_{24}^2\right)\dot{\theta}_1^2$$
$$+ \frac{1}{2}\left(- p_5.d_3. + p_9 + p_4.d_3^2. + 2.p_6..c_4.d_3\right)\dot{\theta}_2^2 + \frac{1}{2}.p_4.\dot{d}_3^2 + \frac{1}{2}.p_{10}.\dot{\theta}_4^2 + \frac{1}{2}.p_{11}.\dot{\theta}_5^2$$
$$+ p_{11}.c_{24}.\dot{\theta}_1.\dot{\theta}_5 - p_6.s_4.\dot{\theta}_2.\dot{d}_3 - p_6.s_4.\dot{d}_3.\dot{\theta}_4 + (p_{10} + p_6.c_4.d_3).\dot{\theta}_2.\dot{\theta}_4 \qquad (2.22)$$

avec:

$p_1 = Izz_1$

$p_2 = Izz_3 + Iyy_2$

$p_3 = m_2.y_2^2 + m_3.z_3^2 + Ixx_2 + Ixx_3.$

$p_7 = Izz_5 + Iyy_4$

$p_8 = m_4.y_4^2 + m_5.z_5^2 + Ixx_4 + Ixx_5.$

$p_9 = m_2.y_2^2 + m_3.z_3^2 + m_4.y_4^2 + m_5.z_5^2 + Ixx_5 + Iyy_3. + Izz_2 + Izz_4$

$p_{10} = m_4.y_2^2 + m_5.z_5^2 + Izz_4 + Ixx_5.$

$p_{11} = Izz_5$

Alors le Lagrangien [Craig, 1986] nous donne les équations dynamiques décrivant le mouvement du robot sous la forme suivante :

$$\tau_1 = M_{11}.\ddot{\theta}_1 + M_{15}.\ddot{\theta}_5 + B_{11}\dot{\theta}_1\dot{\theta}_2 + B_{12}\dot{\theta}_1\dot{d}_3 + B_{13}\dot{\theta}_1\dot{\theta}_4$$
$$+ B_{17}\dot{\theta}_2\dot{\theta}_5 + B_{17}\dot{\theta}_4\dot{\theta}_5 + G_1 + F_1\dot{\theta}_1. \qquad (2.23)$$

$$\tau_2 = M_{22}.\ddot{\theta}_2 + M_{23}\ddot{d}_3 + M_{24}.\ddot{\theta}_4 - B_{17}\dot{\theta}_1\dot{\theta}_5 + B_{25}\dot{\theta}_2\dot{d}_3 + B_{26}\dot{\theta}_2\dot{\theta}_4$$
$$- \tfrac{1}{2}B_{11}\dot{\theta}_1^2 + B_{26}\dot{\theta}_4^2 + G_2 + F_2\dot{\theta}_2. \qquad (2.24)$$

$$\tau_3 = M_{23}.\ddot{\theta}_2 + M_{33}\ddot{d}_3 + M_{23}.\ddot{\theta}_4 + B_{36}\dot{\theta}_2\dot{\theta}_4 - \tfrac{1}{2}B_{12}\dot{\theta}_1^2$$
$$- \tfrac{1}{2}B_{25}\dot{\theta}_2^2 + \tfrac{1}{2}B_{36}\dot{\theta}_4^2 + G_3 + F_3\dot{\theta}_3. \qquad (2.25)$$

$$\tau_4 = M_{24}.\ddot{\theta}_2 + M_{23}\ddot{d}_3 + M_{44}.\ddot{\theta}_4 - B_{36}\dot{\theta}_2\dot{d}_3 - B_{17}\dot{\theta}_1\dot{\theta}_5$$
$$- \tfrac{1}{2}B_{13}\dot{\theta}_1^2 - \tfrac{1}{2}B_{26}\dot{\theta}_2^2 + G_4 + F_4\dot{\theta}_4. \qquad (2.26)$$

$$\tau_5 = M_{11}.\ddot{\theta}_1 + M_{55}.\ddot{\theta}_5 + B_{17}\dot{\theta}_1\dot{\theta}_2 + B_{17}\dot{\theta}_1\dot{\theta}_4 + F_5\dot{\theta}_5. \qquad (2.27)$$

De façon générale, on écrit ce modèle dynamique sous sa forme traditionnelle comme suit :

$$\tau(\theta) = M(\theta).\ddot{\theta} + V_m(\theta,\dot{\theta}).\dot{\theta} + G(\theta) + F(\dot{\theta}).\dot{\theta} \tag{2.28}$$

Puis on déduit ses composantes comme ci-dessous :

$$M(\theta) = \begin{bmatrix} M_{11} & 0 & 0 & 0 & M_{15} \\ 0 & M_{22} & M_{23} & M_{24} & 0 \\ 0 & M_{23} & M_{33} & M_{23} & 0 \\ 0 & M_{24} & M_{23} & M_{44} & 0 \\ M_{15} & 0 & 0 & 0 & M_{55} \end{bmatrix} \tag{2.29}$$

$$V_m(\theta,\dot{\theta}) = \frac{1}{2}\begin{bmatrix} B_{11}.\dot{\theta}_2 + B_{12}.\dot{d}_3 + B_{13}.\dot{\theta}_4 & B_{11}.\dot{\theta}_1 + B_{17}.\dot{\theta}_5 \\ -(B_{11}.\dot{\theta}_1 + B_{17}.\dot{\theta}_5) & B_{23}.\dot{d}_3 + B_{26}.\dot{\theta}_4 \\ -B_{12}.\dot{\theta}_1 & B_{36}.\dot{\theta}_4 - B_{25}.\dot{\theta}_2 \\ -(B_{13}.\dot{\theta}_1 + B_{17}.\dot{\theta}_5) & -(B_{36}.\dot{d}_3 + B_{26}.\dot{\theta}_2) \\ B_{17}.\dot{\theta}_2 + B_{17}.\dot{\theta}_4 & B_{17}.\dot{\theta}_1 \end{bmatrix}$$
$$\begin{bmatrix} B_{12}.\dot{\theta}_1 & B_{13}.\dot{\theta}_1 + B_{17}.\dot{\theta}_5 & B_{17}.(\dot{\theta}_2 + \dot{\theta}_4) \\ B_{25}.\dot{\theta}_2 & B_{26}.(\dot{\theta}_2 + \dot{\theta}_4) & -B_{17}.\dot{\theta}_1 \\ 0 & B_{36}.(\dot{\theta}_2 + \dot{\theta}_4) & 0 \\ -B_{36}.\dot{\theta}_2 & 0 & -B_{17}.\dot{\theta}_1 \\ 0 & B_{17}.\dot{\theta}_1 & 0 \end{bmatrix} \tag{2.30}$$

$$G(\theta) = \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \\ 0 \end{bmatrix} \tag{2.31}$$

$$F(\theta) = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{bmatrix} \tag{2.32}$$

avec :

$$M_{11} = \left(- p_5.d_3.s_2^2 + p_4.d_3^2.s_2^2 + 2.p_6.s_2.s_{24}.d_3 + p_2.c_2^2 + p_7.c_{24}^2 + p_1 + p_3.s_2^2 + p_8.s_{24}^2\right)$$

$$M_{15} = M_{51} = p_{11}.c_{24}$$

$$M_{22} = \left(- p_5.d_3 + p_9 + p_4.d_3^2 + 2.p_6..c_4.d_3\right)$$

$$M_{23} = M_{32} = M_{34} = M_{43} = -p_6..s_4$$

$$M_{24} = M_{42} = (p_{10} + p_6.c_4.d_3)$$

$$M_{33} = p_4$$

$$M_{44} = p_{10}$$

$$M_{55} = p_{11}$$

et aussi :

$$B_{11} = 2\left(- c_2.s_2.d_3.p_5 + p_4.d_3^2.c_2.s_2 + p_6.(c_2.s_{24} + s_2.c_{24}).d_3 - p_2.c_2.s_2 - p_7.s_{24}.c_{24} + p_3.c_2.s_2 + p_8.c_{24}.s_{24}\right)$$

$$B_{12} = 2\left(- 0.5.p_5.s_2^2 + p_4.d_3.s_2^2 + p_6.s_2.s_{24}\right)p_2.c_2^2 + p_7.c_{24}^2 + p_1 + p_3.s_2^2 + p_8.s$$

$$B_{13} = 2\left(p_8.c_{24}.s_{24} - p_7.c_{24}.s_{24} + p_6..s_2.c_{24}\right)$$

$$B_{17} = -p_{11}..s_{24}$$

$$B_{25} = 2\left(- 0.5.p_5 + p_4.d_3 + p_6.c_4\right)$$

$$B_{26} = -2\left(p_6.s_4.d_3\right)$$

$$B_{36} = -2\left(p_6.c_4\right)$$

$$G_1 = G_5 = 0$$

$$G_2 = -g\left(p_{12}.s_2 - 0.5.p_5.s_2 + p_6..s_{24} + p_4.s_2.d_3\right)$$

$$G_3 = g\left(p_4.c_2\right) \qquad G_4 = g\left(p_6.s_{24}\right)$$

$$F = \begin{bmatrix} F_1 & F_2 & F_3 & F_4 & F_5 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 \end{bmatrix}$$

2.3 Génération de la trajectoire

2.3.1 Méthode de la déviation bornée

Dans la plupart des applications on exige que le robot suive une trajectoire rectiligne dans l'espace de travail. Pour ce faire, on doit calculer les trajectoires dans l'espace des articulations qui assurent le mouvement rectiligne dans l'espace de travail.
Pour contrôler les n articulations d'un robot, la trajectoire désirée doit être spécifiée dans l'espace des articulations. La position de départ, celle d'arrivée, ainsi que l'orientation du poignet du robot sont supposées être données dans l'espace cartésien sous forme de deux points p_0 et p_f à l'instant de départ t_0 et à l'instant d'arrivée ou finale t_f respectivement. Il faut convertir ces deux positions « cartésiennes » à leurs équivalents dans l'espace articulaire : q_0 et q_f respectivement. Lorsqu'on transforme la ligne droite qui relie deux points dans l'espace cartésien (ou espace de travail) à l'espace des articulations, cette transformation ne résulte pas nécessairement une ligne droite mais plutôt une courbe [Dessaint et al, 1992].

La méthode de la déviation bornée (Bounded Deviation Method) qui est proposée par [Taylor, 1979] nous permet de déterminer l'ensemble des points, dans l'espace des articulations, qui assure une trajectoire rectiligne dans l'espace de travail tout en gardant un certain intervalle d'approximation.

2.3.2 Principe de la méthode

Cette méthode consiste à remplacer la trajectoire rectiligne par un ensemble de points. Cet ensemble contient, essentiellement, le point de départ et le point d'arrivée et plusieurs points entre eux qui sont désignés par les points intermédiaires. Pour les

trajectoires rectilignes dans l'espace de travail, on doit déterminer les points intermédiaires nécessaires dans l'espace articulaire qui nous permettent de suivre la trajectoire rectiligne dans l'espace de travail. La base de cette méthode est de maintenir la déviation entre la trajectoire désirée et la trajectoire approximative dans l'espace de travail à l'intérieur d'un certain intervalle (Figure 5). Plus que l'intervalle de déviation permise est plus restreint, plus que le nombre des points intermédiaires augmente pour assurer une plus haute précision.

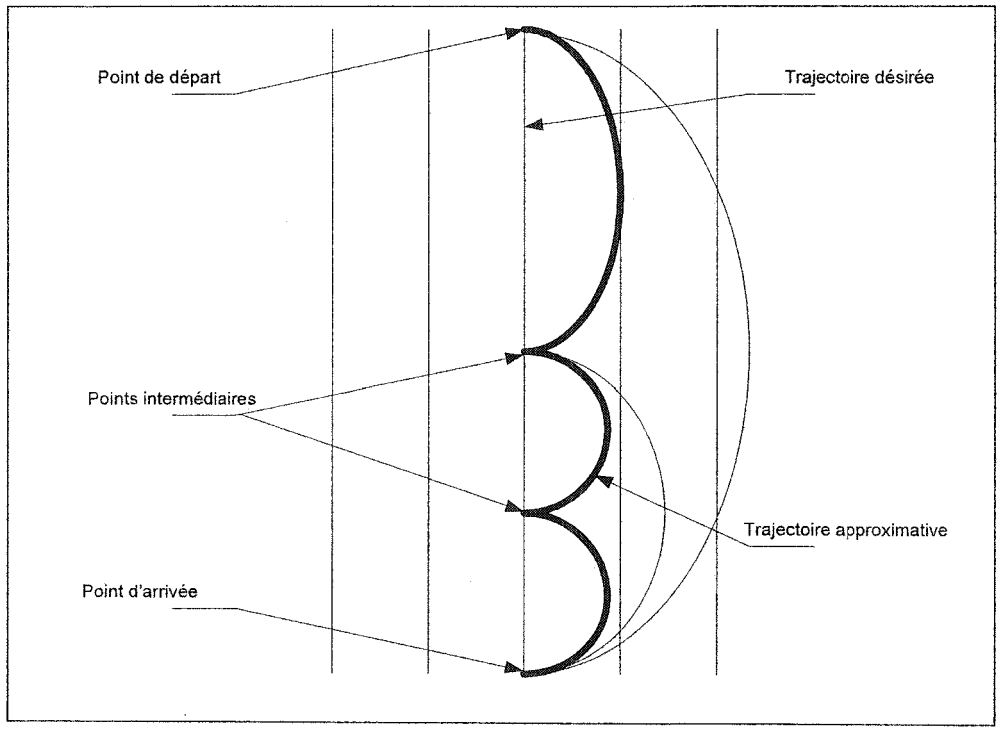


Figure 5 Découpage d'un segment de trajectoire

2.3.3 Algorithme de la méthode

Supposons qu'on veut partir, dans l'espace de travail, du point initial A au point final B moyennant des trajectoires polynomiales de troisième ordre au niveau des articulations. La déviation maximale dans l'espace de travail correspondant, approximativement, au milieu de la trajectoire articulaire. Par la suite, La déviation (ou l'erreur) est examinée à ce point et si elle dépasse la tolérance exigée, ce point est ajouté comme un point intermédiaire. Puis, le test est répété, d'une façon récursive, pour les deux nouveaux segments jusqu'à ce que toutes les déviations soient à l'intérieur de l'intervalle d'approximation permise. L'algorithme 2.1 résume cette méthode.

Algorithme 2.1 de Taylor:

1. Choisir la largeur de l'intervalle de déviation : $\epsilon > 0$;
2. Utiliser la cinématique inverse pour calculer les vecteurs (q_0, q_1) dans l'espace des articulations associées avec les points (p_0, p_1) dans l'espace de travail;
3. Calculer le milieu dans l'espace des articulations : $q_m = \frac{1}{2}(q_0 + q_1)$;
4. Calculer le point p_m associé au vecteur q_m en utilisant les équations de la cinématique directe;
5. Calculer le milieu du segment constitué par les deux points (p_0, p_1) dans l'espace de travail : $p_M = \frac{1}{2}(p_0 + p_1)$.
6. Si la déviation $| p_m - p_M | < \epsilon$, donc, arrêter; sinon, ajouter le point p_M comme point intermédiaire entre p_0 et p_1. Puis, appliquer, récursivement, les étapes 3 à 6 aux deux nouveaux segments (p_0, p_M) et (p_M, p_1)

2.3.4 Technique d'interpolation

La technique d'interpolation consiste à interpoler entre les points intermédiaires pour produire une trajectoire souple (ou lisse) c'est à dire qu'elle admet au moins deux dérivées continues pour éviter le cas d'une accélération infinie. Pour satisfaire à cette exigence, on peut adopter les trajectoires cubiques. Le détail de calcul est reporté en annexe. Mais, pour illustrer la performance du planificateur de trajectoires, la partie suivante traite un exemple en utilisant la cinématique du robot MAKER 100.

2.3.5 Trajectoire triangulaire

Dans cette partie, on cherche à déterminer dans le domaine articulaire les courbes des positions, les vitesses et les accélérations articulaires des cinq articulations du robot MAKER 100 en fonction du temps, lorsqu'on fixe comme trajectoire un triangle dans l'espace cartésien dont les sommets sont regroupés dans le tableau IV.

Tableau IV

Coordonnées de l'effecteur du robot aux sommets d'un triangle (consigne 1)

	X (m)	Y (m)	Z (m)	roulis (rd)	tangage (rd)
Point A	0.4	0.4	0.4	0	0
Point B	0.6	0	0	$-\pi/2$	$+\pi/2$
Point C	0	0.6	0	$-\pi/4$	π

De plus, en faisant augmenter la valeur l_5 relative à la longueur du lien 5 du robot, on constate que le nombre des points nécessaires diminue comme le montre le tableau V suivant :

Tableau V

Nombre de points nécessaires pour une génération de trajectoires

l_5 (m)	Nombre de points nécessaires	
	Tolérance=0.001	Tolérance=0.01
0	48	16
0.03	34	10
0.05	34	10
0.10	34	10
0.15	34	10
0.20	27	10
0.25	18	6

Ensuite, pour bien illustrer cet algorithme, on a déterminé l'évolution temporelle dans l'espace des articulations du vecteur de position, de vitesse et d'accélération pour une durée de 10 secondes; et en prenant l_5=0.

Les figures 6 à 10 montrent l'évolution de la position, de la vitesse et de l'accélération des cinq articulations du robot.

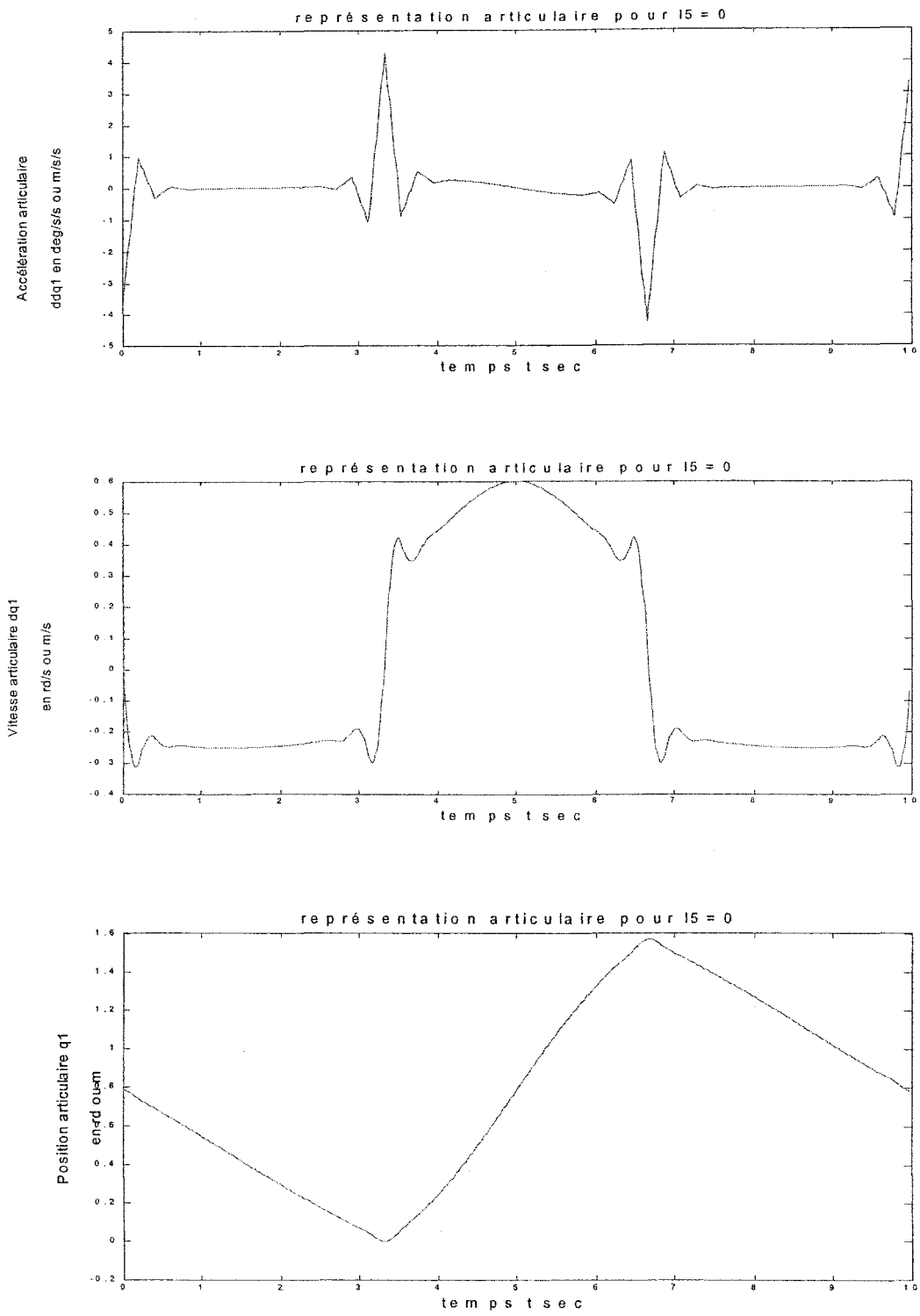


Figure 6 La position, la vitesse et l'accélération de l'articulation #1

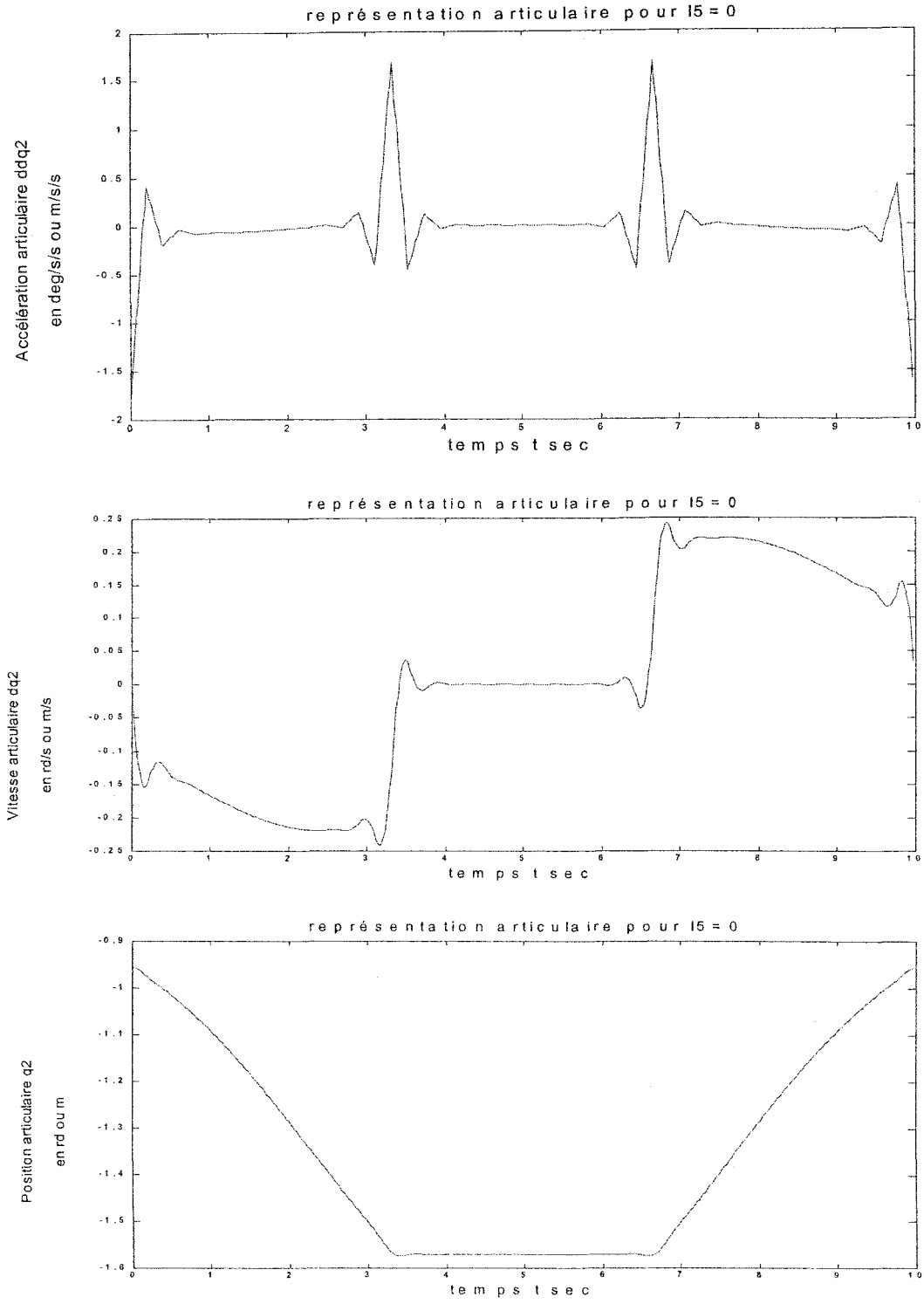


Figure 7 La position, la vitesse et l'accélération de l'articulation #2

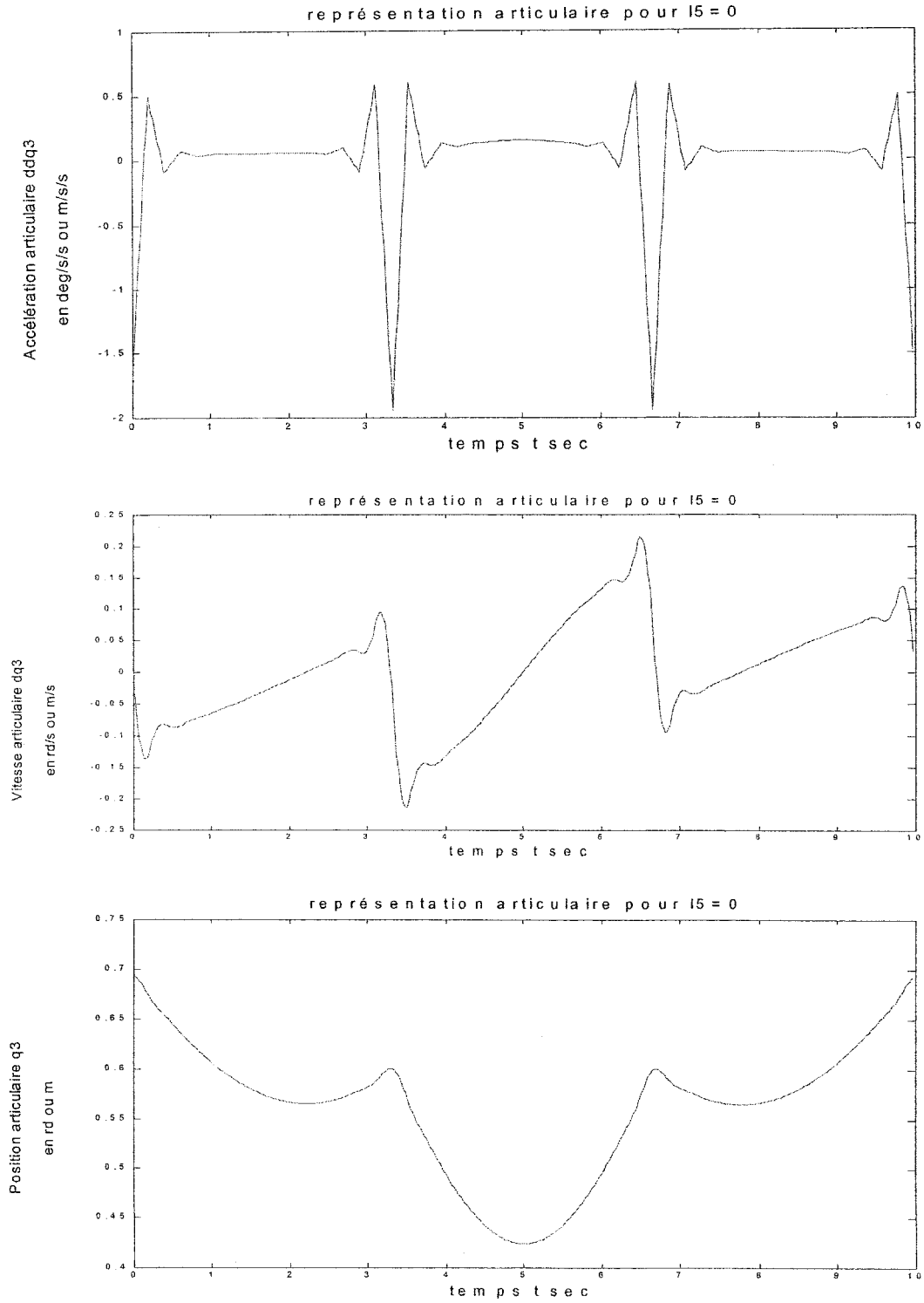


Figure 8 La position, la vitesse et l'accélération de l'articulation #3

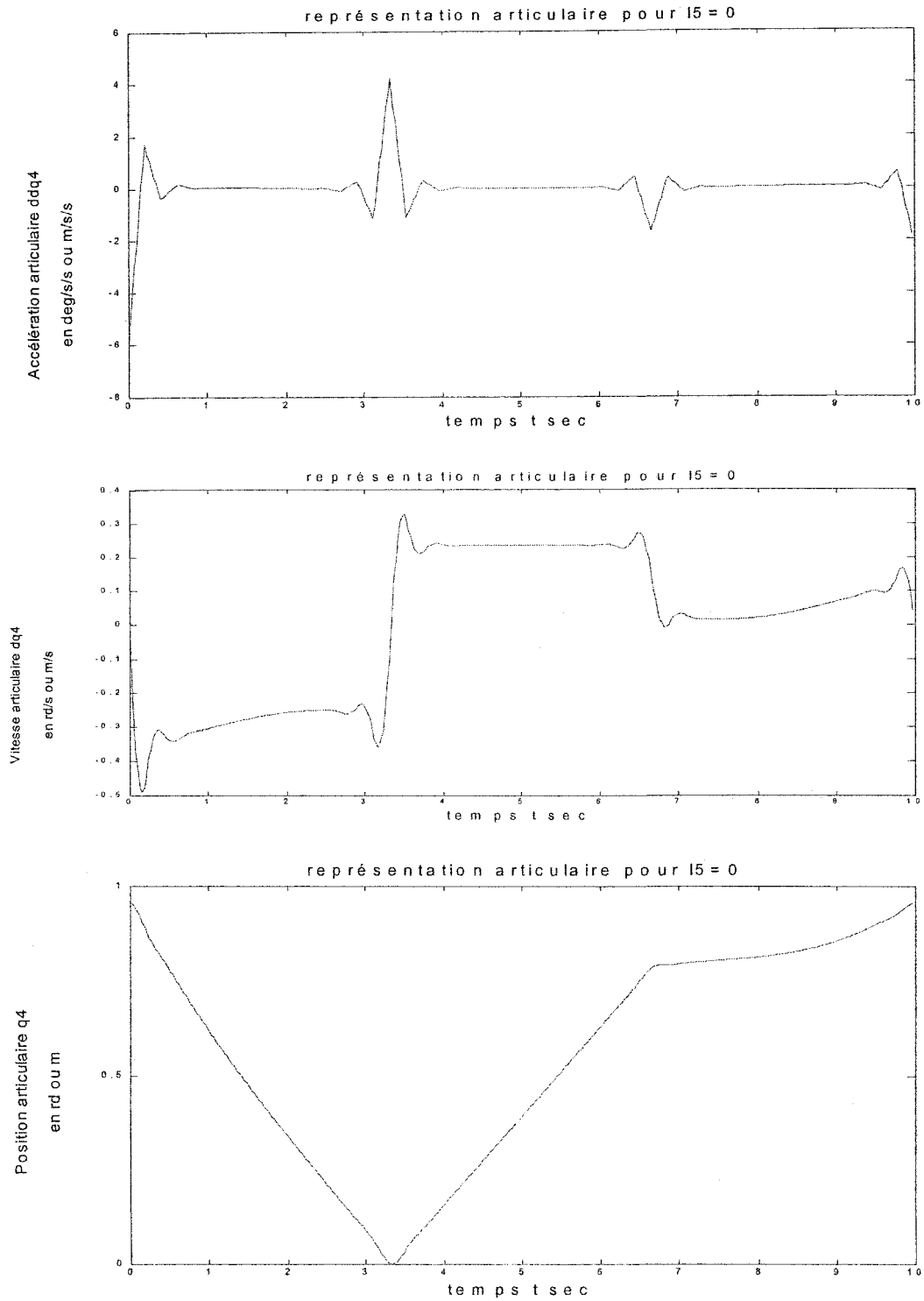


Figure 9 La position, la vitesse et l'accélération de l'articulation #4

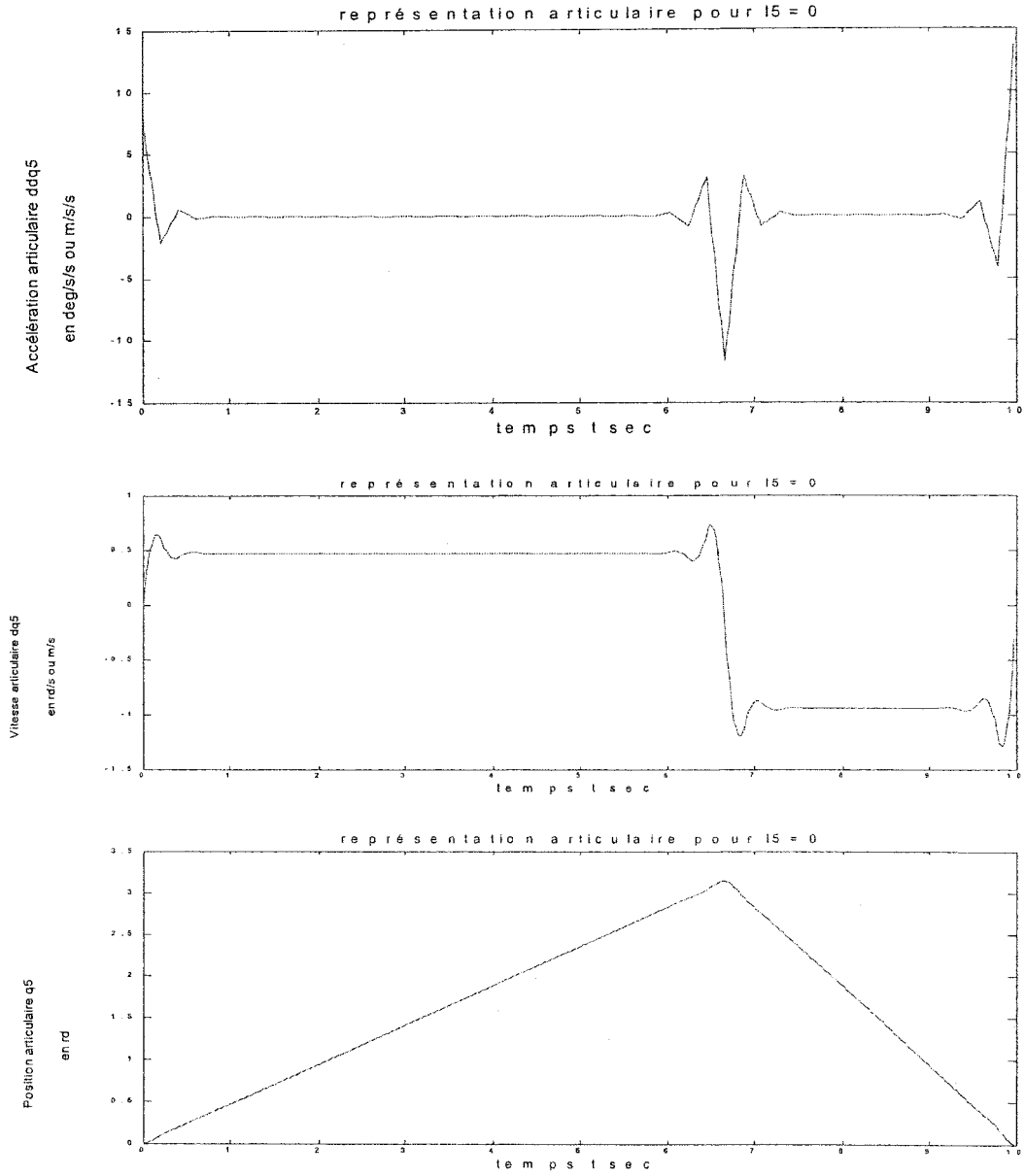


Figure 10 La position, la vitesse et l'accélération de l'articulation #5

De plus, nous avons exprimé les points, nécessaires pour la génération de la trajectoire cartésienne décrite par le triangle mentionné plus haut, dans le repère cartésien. La figure 11 illustre les points intermédiaires nécessaires.

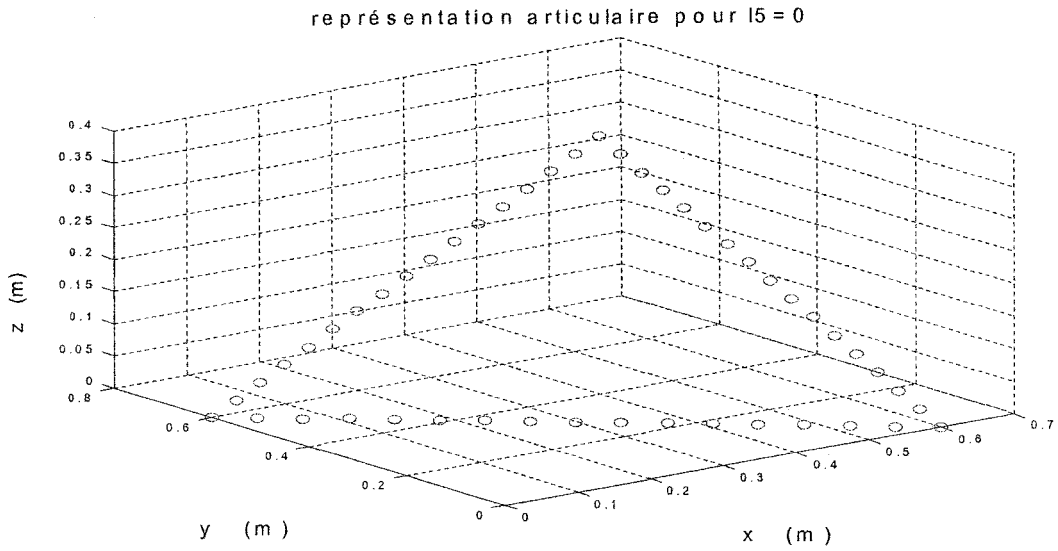


Figure 11 Les points nécessaires pour la génération.

Après avoir déterminer la cinématique directe et inverse du robot US MAKER 100, nous avons modélisé la dynamique du robot par la méthode de Lagrange Euler, et par la suite, nous avons généré la trajectoire articulaire qui sera comme consigne de notre commande qui est l'objet du chapitre suivant.

CHAPITRE 3

LA COMMANDE ADAPTATIVE APPLIQUÉE AUX ROBOTS

Le présent chapitre est consacré à la commande adaptative du robot manipulateur dans ses coordonnées articulaires. Cette commande suppose que les positions et les vitesses articulaires soient mesurables. On considère que la structure utilisée est bien connue. D'après le chapitre précédant, elle tient compte des effets de frottements visqueux, mais tout effet d'élasticité dans les articulations est négligé ainsi que la flexibilité de la chaîne cinématique. Les méthodologies de commande adaptative proposées s'intéressent aux incertitudes paramétriques du modèle tout en supposant que ces paramètres sont constants.

3.1 Robot manipulateur généralisé

Les robots manipulateurs sont des systèmes mécaniques articulés formés par des corps rigides reliés entre eux au moyen d'articulations. Dans le présent projet, nous considérerons un robot manipulateur constitué par une chaîne cinématique ouverte comme celle qui est montrée de façon abstraite sur la figure 12.

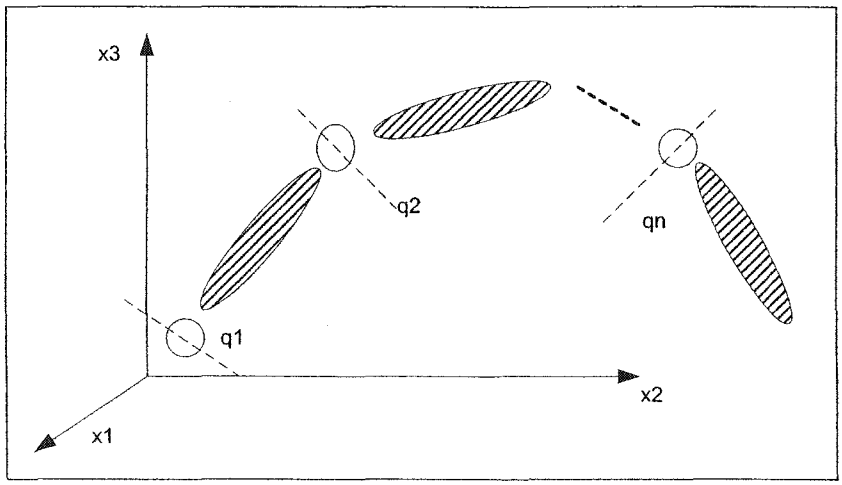


Figure 12 Diagramme abstrait d'un robot manipulateur de n degrés de liberté

Les positions articulaires correspondantes à chaque articulation du robot, qui sont mesurées au moyen de capteurs placés sur les actionneurs, sont regroupées dans le vecteur de positions articulaires q. Le comportement dynamique des robots manipulateurs peut être décrit au moyen d'équations différentielles ordinaires qui appartiennent à la catégorie des systèmes dynamiques appelés Euler-Lagrange. Tout modèle utile à des fins de contrôle de systèmes réels possède divers paramètres en rapport avec les caractéristiques physiques de celui-ci. Le modèle des robots manipulateurs contient aussi des paramètres dépendants de leurs caractéristiques mécaniques; cependant, certains de ces paramètres peuvent être incertains, c'est-à-dire que leurs valeurs numériques ne sont connues qu'avec une très grande incertitude. Ces paramètres incertains sont habituellement, associés à des caractéristiques du robot comme l'inertie et les distances aux centres de masse.

À cause de la grande variété d'applications pour les robots manipulateurs, ceux-ci représentent un terrain fertile pour la formulation de divers problèmes de commande. Les objectifs de commande pour des manipulateurs robotiques, recouvrent un vaste spectre allant de la commande de position pure (régulation), du mouvement (poursuite), de la force, de l'impédance mécanique jusqu'à la commande hybride position/force.

La complexité dans la formulation de la commande de robots peut être altérée par les hypothèses considérées, pour l'obtention du modèle du robot. Parmi ces considérations, surgissent principalement, la présence ou l'absence de frottement dans les articulations, l'élasticité dans les articulations et la flexibilité de chaque membre de la chaîne cinématique.

3.2 Linéarité par rapport aux paramètres dynamiques

Une des caractéristiques des robots manipulateurs, qui s'avère vitale, pour la synthèse des contrôleurs adaptatifs, est que: leurs modèles dynamiques peuvent s'exprimer en termes linéaires d'un ensemble de paramètres correctement choisis appelés paramètres

dynamiques. Les paramètres dynamiques dépendent des masses, inerties et positions des centres de masse de chaque membre de la chaîne cinématique et de l'objet manipulé par le robot, celui-ci pouvant être considéré comme la partie du chaînon final. Cette propriété des robots manipulateurs a été rapportée par [Lozano et Taoutaou, 2001].
En effet, étant donnés $u, v, w \in R^n$, le modèle dynamique d'un robot de n degrés de liberté satisfait :

$$M(q,p).u + Vm(q,w,p).v + G(q,p) = W(q,u,v,w).p + k(q,u,v,w) \tag{3.1}$$

Avec $p \in R^m$, $k(q,u,v,w) \in R^n$ et $W(q,u,v,w) \in R^{nxm}$ étant la matrice de régression. Le vecteur p, connu sous le nom de vecteurs de paramètres dynamiques, contient des éléments qui dépendent de paramètres du manipulateur et de l'objet manipulé.

Étant donnée une sélection de masse, inerties et distances aux centres de masse, nous déterminons par la suite les paramètres dynamiques à partir du modèle du robot. Des procédures pour obtenir les composants du modèle de régression, sont présentées dans Lozano, [2001]. Naturellement, il sera toujours possible de choisir un vecteur de paramètres dynamiques p pour satisfaire (3.1) avec $k(q,u,v,w) = 0 \in R^n$.

Par la suite le modèle dynamique va être écrit de la manière suivante :

$$M(q,p).u + Vm(q,w,p).v + G(q,p) = \tau \tag{3.2}$$

Avec cette nouvelle notation, la paramétrisation pourra s'exprimer comme suit:

$$M(q,p).u + Vm(q,w,p).v + G(q,p) = W(q,u,v,w).p + M_0(q).u + Vm_0(q,w).v + G_0(q) \tag{3.3}$$

où on peut identifier :

$$k(q,u,v,w) = M_0(q).u + Vm_0(q,w).v + G_0(q)$$

Les matrices M_0, Vm_0 et le vecteur G_0 présentent les parties des matrices, et du vecteur de gravité, qui ne dépendent pas du vecteur de paramètres dynamiques respectivement.

3.3 Identification paramétrique

On peut distinguer quelques méthodes pour déterminer les paramètres dynamiques d'un robot manipulateur en utilisant les données des modèles des membres de la chaîne cinématique; par la mesure de signaux pendant les mouvements du robot. Parmi cette grande variété de procédures d'identification, nous abordons celle de « régression par dynamique filtré ».

De plus, le modèle dynamique des robots, peut être écrit de la manière suivante :

$$\tau = W(q,\dot{q},\dot{q},\ddot{q}).p \tag{3.4}$$

où $W(q,\dot{q},\dot{q},\ddot{q})$ est la matrice de régression de n x m qui dépend des fonctions connues.

Notons que pour calculer les éléments de W, on a besoin de mesurer l'accélération des articulations q. Pour surmonter cet obstacle, Hsu, et Al [Lozano et Taoutaou, 2001], ont proposé de filtrer les deux côtés du modèle du robot avec un filtre approprié strictement stable de type $f(s) = \frac{\lambda}{s+\lambda}$, où $\lambda > 0$ et s correspond à l'opérateur différentiel. Par conséquent, le modèle de régression par dynamique filtrée peut s'écrire de la manière suivante :

$$\tau_f = W_f(q,\dot{q}).p \tag{3.5}$$

Où :

$\tau_f = f(s).\tau$

$W_f = f(s).W(q,\dot{q})$

Cette technique est souvent utilisée pour la commande adaptative indirecte appliquée aux robots et aussi pour l'identification des paramètres dynamiques du robot.

3.4 Commande adaptative du mouvement

Considérons l'équation dynamique pour des robots de n degrés de liberté en tenant compte de la paramétrisation linéaire :

$$\begin{aligned} &M(q,p).u + Vm(q,w,p).v + G(q,p) = \tau \\ &= W(q,u,v,w).p + M_0(q).u + Vm_0(q,w).v + G_0(q) \end{aligned} \tag{3.6}$$

En général, un contrôleur adaptatif est formé de deux parties :

- la loi de commande du contrôleur,
- la loi d'adaptation.

La loi de commande est, en général, une équation algébrique (mais souvent dynamique) servant à calculer l'action de contrôle qui peut s'exprimer de manière fonctionnelle ainsi :

$$\tau = \tau(t,\ q,\ \dot{q}, q_d, \dot{q}_d, \ddot{q}_d, \hat{p}).$$

On connaît le vecteur $\hat{p} \in R^m$ sous le nom de vecteur de paramètres adaptatifs. La loi d'adaptation permet de déterminer $\hat{p}(t)$, et en général, elle peut s'exprimer comme une équation différentielle en $\hat{p} \in R^m$:

$$\hat{p}(t) = \Gamma . \int_0^t \Psi(r,\ q,\ \dot{q}, q_d, \dot{q}_d, \ddot{q}_d) dr + \hat{p}(0) \tag{3.7}$$

Où $\Gamma \in R^{mxm}$ est normalement diagonale et définie positive et est nommé « gain d'adaptation ». $\hat{p}(0)$ est un vecteur arbitraire, bien que dans la pratique, il soit choisi comme la meilleure approximation à porter du vecteur de paramètres dynamiques

$\hat{p} \in R^m$. Ψ est une fonction vectorielle de dimension m à déterminer pour que le système soit plus stable.

L'ampleur du gain d'adaptation est en relation, avec la vitesse d'adaptation du système de contrôle, face à l'incertitude paramétrique du modèle dynamique du robot. Néanmoins, dans la pratique, celui-ci s'obtient par des essais jusqu'à obtenir un comportement satisfaisant du système de contrôle.

La figure 13 montre un schéma bloc de la commande adaptative d'un robot.

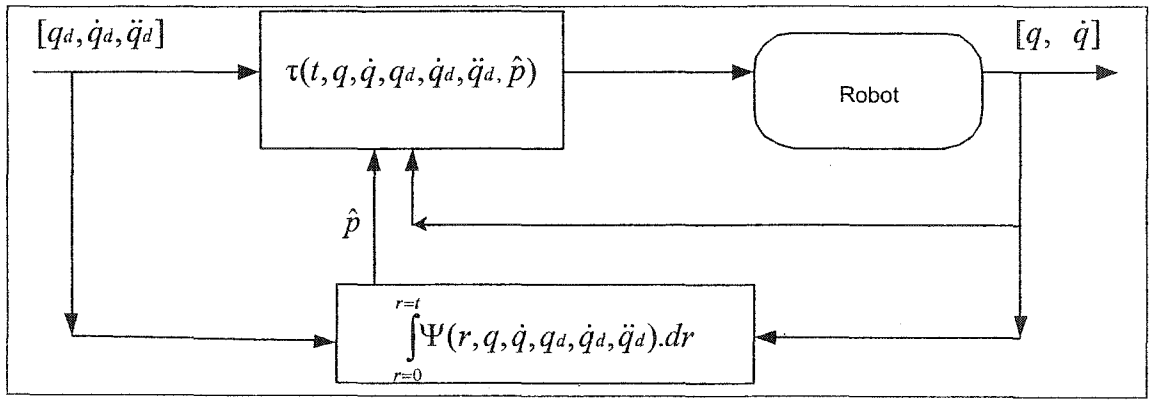


Figure 13 Commande adaptative de robots

3.5 Commande adaptative directe [Slotine et Li, 1987]

C'est une technique très populaire dans le milieu scientifique depuis 1987, portant le nom de ses créateurs. Dans sa version adaptative, le contrôleur de Slotine et Li peut s'exprimer au moyen de la loi de commande suivante :

$$\begin{aligned} \tau &= M(q,\hat{p}).u + Vm(q,w,\hat{p}).v + G(q,\hat{p}) + Kv.\dot{\tilde{q}} + kp.\tilde{q} \\ &= W(q,u,v,w).\hat{p} + M_0(q).u + Vm_0(q,w).v + G_0(q) + Kv.e \end{aligned} \tag{3.8}$$

avec :

$$\dot{\tilde{q}} = \dot{q}_d - \dot{q}.$$

$$\tilde{q} = q_d - q.$$

$$\Lambda = Kv^{-1}.Kp.$$

$$e = \dot{\tilde{q}} + \Lambda.\tilde{q}.$$

$$u = \ddot{q}_d + \Lambda.\dot{\tilde{q}}.$$

$$v = \dot{q}_d + \Lambda.\tilde{q}.$$

$$w = \dot{q}.$$

$$\frac{d\hat{p}}{dt} = \Gamma.W^T.e.$$

Où $Kv, Kp \in R^{nxn}$ sont des matrices symétriques définies positives, $\tilde{q}$ révèle l'erreur de position.

La figure 14 présente le diagramme de blocs correspondant à la commande adaptative de Slotine et Li.

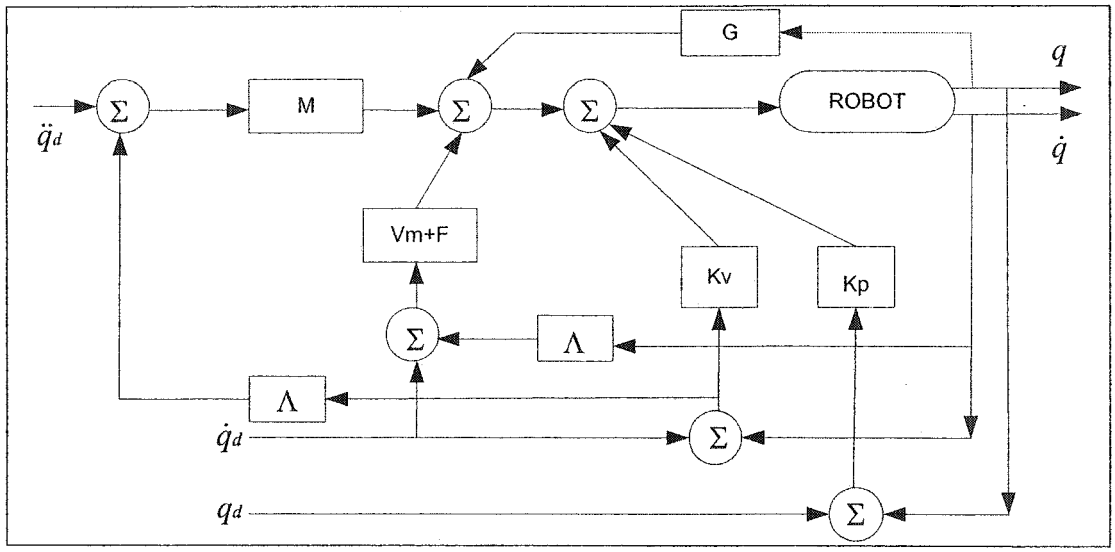


Figure 14 Commande adaptative directe

3.6 Étude de la stabilité du contrôleur

En intégrant la loi de commande de Slotine et Li avec les équations du modèle dynamique du robot, on aboutit [Lozano et Taoutaou, 2001] au système dynamique suivant :

$$\frac{d}{dt}\begin{bmatrix}\tilde{q}\\ \dot{\tilde{q}}\\ \hat{p}\end{bmatrix}=\begin{bmatrix}\dot{\tilde{q}}\\ M(q,p)^{-1}.\left[-Kv.e-W.\tilde{p}-Vm(q,\dot{q},p).e\right]-\Lambda.\dot{\tilde{q}}\\ \Gamma.W^{T}.e\end{bmatrix}$$

$avec$: (3.9)

$\tilde{p}=\hat{p}-p.$

Cette équation est une équation différentielle parfaitement non linéaire de l'espace d'état R^{2n+m} centrée autour de 0 qui est son point d'équilibre.

L'analyse de stabilité de l'équation de boucle fermée est faite en considérant la fonction candidate de Lyapunov suivante [Lozano et Taoutaou, 2001]:

$$V(\tilde{q},\dot{\tilde{q}},\hat{p})=\frac{1}{2}\begin{bmatrix}\tilde{q}\\ \dot{\tilde{q}}\\ \hat{p}\end{bmatrix}^{T}\begin{bmatrix}2.Kp+\Lambda^{T}.M(q,p).\Lambda & \Lambda^{T}.M(q,p) & 0\\ M(q,p).\Lambda & M(q,p) & 0\\ 0 & 0 & \Gamma^{-1}\end{bmatrix}\begin{bmatrix}\tilde{q}\\ \dot{\tilde{q}}\\ \hat{p}\end{bmatrix} \quad (3.10)$$

$$=\frac{1}{2}.e^{T}.M(q,p).e+\tilde{q}^{T}Kp.\tilde{q}+\frac{1}{2}\hat{p}^{T}.\Gamma^{-1}.\hat{p}.$$

La dérivée temporelle de la fonction candidate de Lyapunov prend la forme suivante après plusieurs simplifications [Lozano et Taoutaou, 2001] :

$$\dot{V}(\tilde{q},\dot{\tilde{q}},\hat{p}) = -\dot{\tilde{q}}^T Kv.\dot{\tilde{q}} - \tilde{q}^T \Lambda^T .Kv.\tilde{q}.$$
$$= -\begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \\ \hat{p} \end{bmatrix}^T \begin{bmatrix} \Lambda^T .Kv.\Lambda & 0 & 0 \\ 0 & Kv & 0 \\ 0 & 0 & \Gamma^{-1} \end{bmatrix} \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \\ \hat{p} \end{bmatrix}. \qquad (3.11)$$

Cela prouve [Lozano et Taoutaou, 2001] que cette fonction candidate de Lyapunov, pour ce système dynamique, est définie positive de manière globale, alors la méthode directe de Lyapunov permet de garantir que l'origine de l'espace d'état du système en boucle fermée est stable et que ses solutions sont bornées.

3.7 Étude de cas du robot MAKER 100.

À partir des équations du modèle dynamique, on fait exprimer la matrice de régression W en fonction des paramètres dynamiques bien choisis, ce qui nous permet d'écrire la matrice W sous la forme suivante :

$W_{11} = \ddot{q}_1.$

$W_{12} = -2.c_2.s_2.\dot{q}_1.\dot{q}_2 + c_2^2\,\ddot{q}_1.$

$W_{22} = s_2.c_2.\dot{q}_1^2.$

$W_{13} = 2.s_2.c_2.\dot{q}_1.\dot{q}_2 + s_2.^2\,\ddot{q}_1.$

$W_{23} = -s_2.c_2.\dot{q}_1^2.$

$W_{14} = 2.s_2.c_2.d_3^2.\dot{q}_1.\dot{q}_2. + 2.s_2^2.d_3.\dot{q}_1.\dot{d}_3. + s_2.^2\,d_3^2.\ddot{q}_1.$

$W_{24} = 2.d_3.\dot{q}_2.\dot{d}_3. - s_2.c_2.d_3^2.\dot{q}_1^2 + d_3^2.\ddot{q}_2. - g.d_3.s_2.$

$W_{34} = -d_3.\dot{q}_2^2 - s_2^2.d_3.\dot{q}_1^2 + \ddot{d}_3.. + g.c_2.$

$$W_{15} = -2.s_2.c_2.d_3.\dot{q}_1.\dot{q}_2. - s_2^2.\dot{q}_1.\dot{d}_3. - s_2^2.d_3.\ddot{q}_1.$$

$$W_{25} = -\dot{q}_1.\dot{d}_3 - d_3.\ddot{q}_2. + s_2.c_2.d_3.\dot{q}_1^2 + \frac{1}{2}g.s_2.$$

$$W_{35} = \frac{1}{2}.s_2^2.\dot{q}_1^2 + \frac{1}{2}.\dot{q}_2^2.$$

$$W_{16} = 2.(s_2.c_{24} + c_2.s_{24}).d_3.\dot{q}_1.\dot{q}_2. + 2.s_2.s_{24}.d_3.\ddot{q}_1. + 2.s_2.c_{24}.d_3.\dot{q}_1.\dot{q}_4. + 2.s_2.s_{24}.\dot{q}_1.\dot{d}_3.$$

$$W_{26} = 2.c_4.\dot{q}_2.\dot{d}_3 - (s_2.c_{24} + c_2.s_{24}).d_3.\dot{q}_1^2 - 2.s_4.d_3.\dot{q}_2.\dot{q}_4 - s_4.d_3.\dot{q}_4^2 + 2.c_4.d_3.\ddot{q}_2 - s_4.\ddot{d}_3 + c_4.d_3.\ddot{q}_4 - g.s_{24.}$$

$$W_{36} = -s_2.s_{24}.\dot{q}_1^2 - c_4.\dot{q}_2^2 - c_4.\dot{q}_4^2 - 2.c_4.\dot{q}_2.\dot{q}_4 - s_4.\ddot{q}_2. - s_4.\ddot{q}_4.$$

$$W_{46} = 2.c_4.\dot{q}_2.\dot{d}_3 + c_4.d_3.\ddot{q}_2. - s_4.\ddot{d}_3 - s_2.c_{24}.d_3.\dot{q}_1^2 + s_4.d_3.\dot{q}_2^2 - g..s_{24.}$$

$$W_{17} = -2.s_{24}.c_{24}.\dot{q}_1.\dot{q}_2 - 2.s_{24}.c_{24}.\dot{q}_1.\dot{q}_4 + c_{24}^2.\ddot{q}_1.$$

$$W_{27} = s_{24}.c_{24}.\dot{q}_1^2.$$

$$W_{47} = s_{24}.c_{24}.\dot{q}_1^2.$$

$$W_{18} = 2.s_{24}.c_{24}.\dot{q}_1.\dot{q}_2. + s_{24}^2.\ddot{q}_1. + 2.s_{24}.c_{24}.d_3.\dot{q}_1.\dot{q}_4.$$

$$W_{28} = -s_{24}.c_{24}.\dot{q}_1^2. \qquad (3.12)$$

$$W_{48} = -s_{24}.c_{24}.\dot{q}_1^2.$$

$$W_{2.9} = \ddot{q}_2.$$

$$W_{2.10} = \ddot{q}_4.$$

$$W_{4.10} = \ddot{q}_2 + \ddot{q}_4.$$

$$W_{1.11} = -s_{24}.\dot{q}_2.\dot{q}_5 - s_{24}.\dot{q}_4.\dot{q}_5. + c_{24}.\ddot{q}_5.$$

$$W_{2.11} = s_{24}.\dot{q}_1.\dot{q}_5.$$

$$W_{4.11} = s_{24}.\dot{q}_1.\dot{q}_5.$$

$$W_{5.11} = -s_{24}.\dot{q}_1.\dot{q}_2. - s_{24}.c_{24}.\dot{q}_1.\dot{q}_4 + c_{24}.\ddot{q}_1. + \ddot{q}_5.$$

$$W_{2.12} = -g.s_2.$$

Au présent chapitre, nous avons rappelé la théorie de la commande adaptative directe selon l'approche de Slotine et Li, qui repose principalement sur un rafraîchissement des paramètres dynamiques estimés par une loi d'adaptation calculée a partir de la matrice de régression du modèle dynamique du robot. Et par la suite, nous avons explicité cette matrice de régression pour le cas du robot US MAKER 100.

CHAPITRE 4

IMPLEMENTATION DU CONTROLEUR EN TEMPS RÉEL

L'énoncé du problème semble plutôt simple : « commander le robot à faire parcourir un chemin sous forme d'un triangle avec une meilleure performance » ; mais le passage de l'étape de la théorie, développée aux chapitres 2 et 3, à l'étape de l'implémentation en temps réel nécessite une bonne connaissance du matériel et une parfaite maîtrise de leurs pilotes afin d'adapter les programmes de simulation aux vrais cas en temps réel.

Ce chapitre décrit le programme basé sur la plate-forme QNX fait avec un environnement graphique et permet une simulation de la commande adaptative directe appliquée au robot MAKER 100 et son implantation en temps réel sur un processeur standard d'Intel placé sur un ordinateur. Le programme, qui comprend les outils de développement et l'interface graphique qui peuvent être exécuté simultanément, affiche une réponse déterministe du système d'exploitation QNX, en adoptant l'architecture récente monoprocesseur. Cette architecture remplace celle traditionnelle de multiprocesseurs, dite hôte/DSP. L'avantage de la première c'est le coût réduit et la grande flexibilité nettement améliorée.

Pour faciliter la compréhension, nous commençons à décrire, brièvement, les deux aspects d'architecture ; et ensuite une description sommaire du matériel implanté en parcourant le design mécanique, électrique et informatique.

4.1 Architecture hôte/DSP

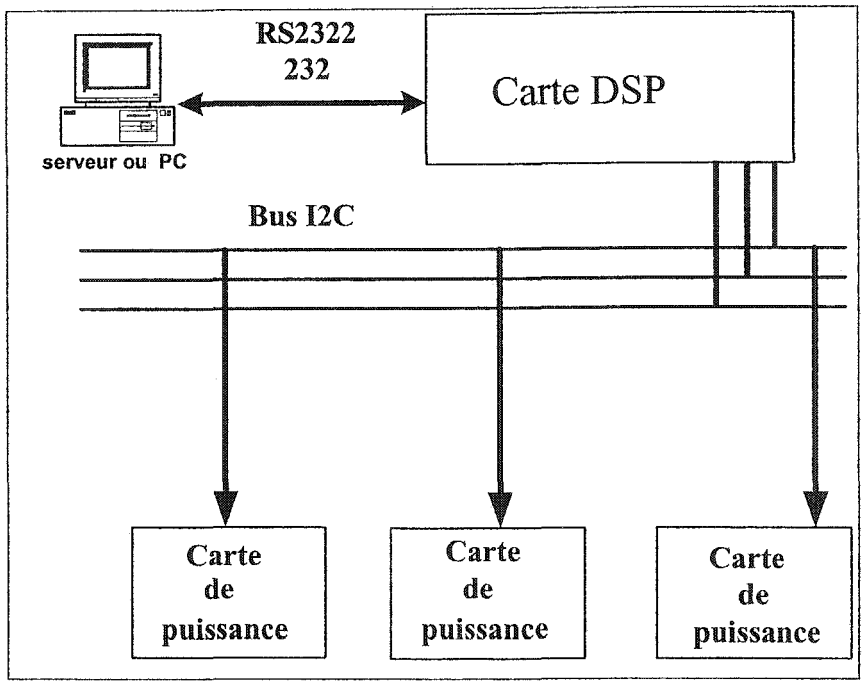


Figure 15 Architecture hôte/DSP

Dans la traditionnelle architecture hôte/DSP, il n'y a pas de contraintes placées sur le système d'exploitation de l'hôte durant l'exécution du programme sur le DSP. La figure 15 montre l'architecture de type hôte/DSP, où l'hôte pourrait être un mac, ou une station de travail d'Unix, ou un ordinateur personnel. Le DSP est utilisé comme interface de l'hôte à travers le bus. Le programme de contrôle est développé, sous l'hôte puis téléchargé dans le DSP où il s'exécute. Les données nécessaires pour le contrôle doivent être transférées rapidement à l'hôte vu la capacité limitée en mémoire vive du DSP (32 K mots). L'interface graphique, si elle existe s'exécute à partir de l'hôte. L'utilisateur peut démarrer, arrêter le programme de contrôle, modifier les gains de contrôle, et améliorer les autres fonctions d'affichage et toutes les connexions Entrée/Sortie à travers le DSP.

Parmi les avantages de la présente topologie [Loffler et al, 1998], il y a la garantie d'une réponse déterministe, contrairement aux systèmes MS-WINDOWS NT, SMP-Linux. Le système hôte/DSP est un exemple des systèmes multiprocesseurs antisymétriques; chaque processeur est dédié à une certaine tâche. Dans ce cas, le DSP exécute seulement le programme de contrôle. Tandis que l'hôte exécute toutes les autres fonctions nécessaires pour une réponse déterministe (affichage de données, interface utilisateur). L'autre avantage de la solution hôte/DSP est que le DSP est désigné par l'exécution rapide des petits programmes qui contiennent des opérations des points flottants.

Toutefois, le présent système présente deux inconvénients majeurs : le coût et la complexité. Le coût du matériel et du logiciel qui augmente avec le nombre de processeurs et aussi avec le degré d'hétérogénéité de l'architecture hôte/DSP en matière de développement. L'utilisateur doit se familiariser avec les deux environnements de développement du DSP et de l'hôte.

4.2 Architecture de type QNX à monoprocesseur

Les pilotes de périphériques installés rendent la programmation la communication avec le matériel, n'est plus nécessaire.

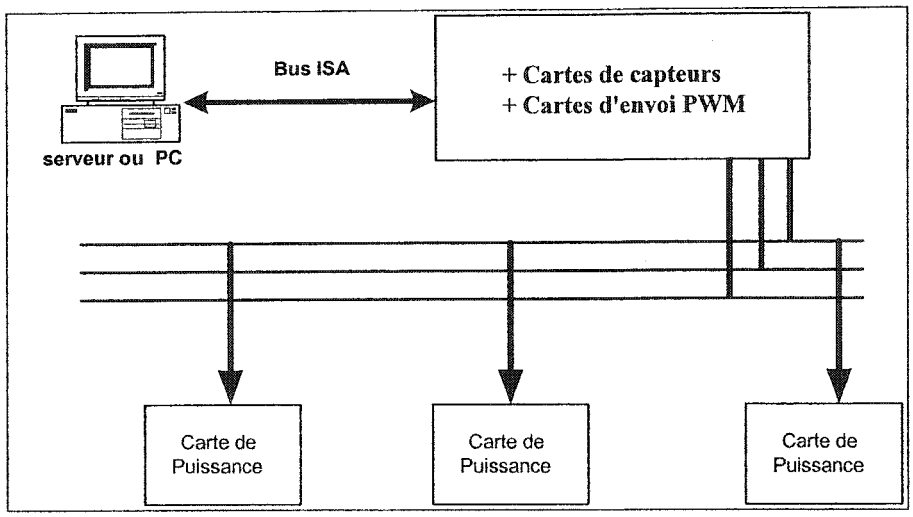


Figure 16 Architecture monoprocesseur avec QNX

En plus des avantages cités plus haut (réponse déterministe, capacité de calcul adéquate), l'avantage de l'actuelle topologie est tout d'abord le dépassement des désavantages de l'ancienne topologie (coût, complexité, flexibilité, actualisation, etc); Et la réponse nettement plus déterministe ainsi que la puissance grandiose de traitement.

Selon les expériences faites jusqu'à présent sur le robot MAKER 100 suivant l'architecture hôte/DSP, le programme de contrôle dans le DSP, consiste à une dizaine de pages de code en langage C. En remplaçant les procédures d'écriture aux canaux A/D, et de lecture des encodeurs, par des fonctions de librairie, cela simplifie la tâche et allége le code (figure 16).

De plus, l'interfaçage du matériel est simplifiée en utilisant des cartes Entrée/Sortie standards qui fournissent A/D, D/A, entrée encodeurs, Entrée/Sortie digitale, horloges.

4.3 Design mécanique

La surface totale occupée par les robots et les accessoires est de 90 pieds carrés environ. Elle comprend les deux structures mécaniques composées des bras de robots, les

moteurs à courant continu pour l'entraînement des bras, l'armoire d'alimentation électrique et les postes de travail ainsi qu'un ensemble de câblage adapté aux besoins.

Le robot que nous avons utilisé était équipé d'une pince à commande pneumatique. Cette pince a deux états stables: ouvert ou fermé (pas de position intermédiaire). L'air comprimé nécessaire pour commander cette pince, est fourni par un compresseur.

4.4 Design électrique

La figure 17 montre la topologie du matériel implanté et les signaux circulants entre le PC et le robot.

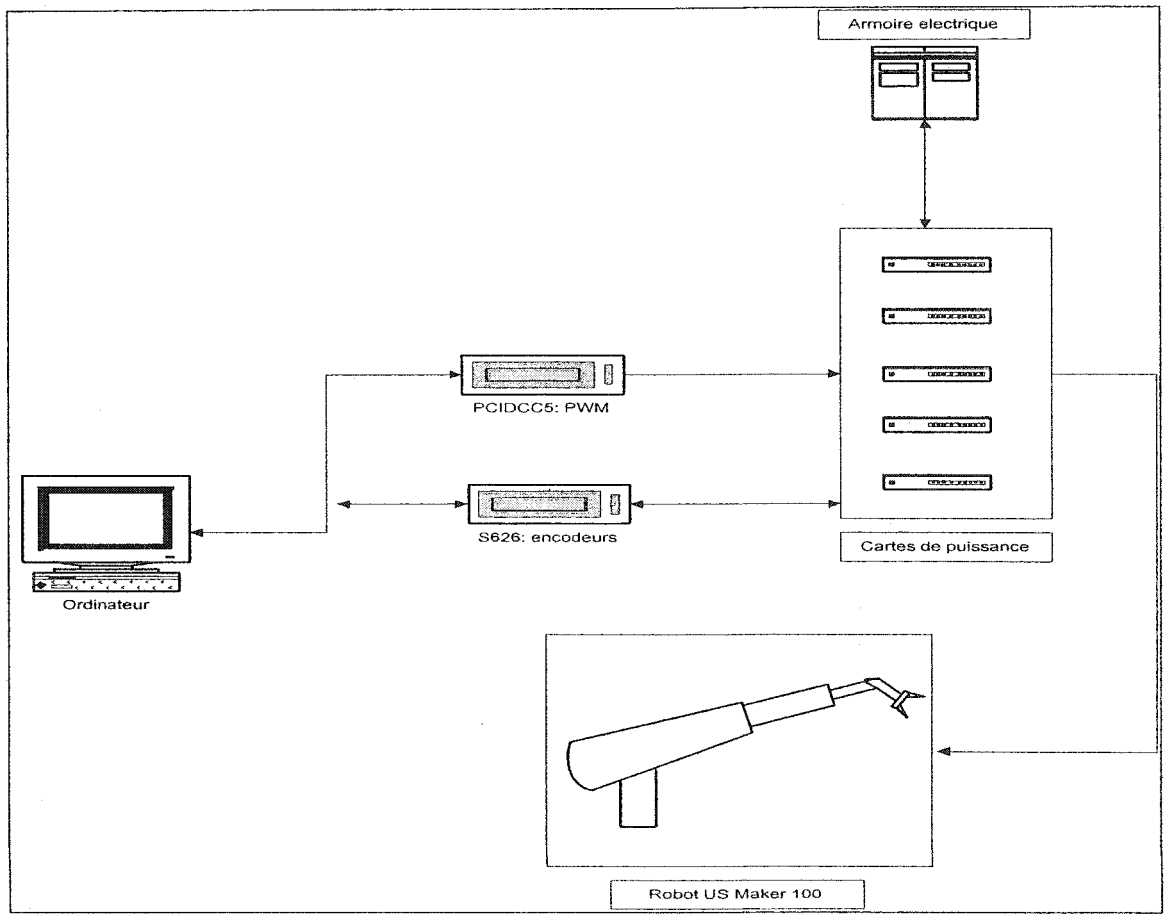


Figure 17 Topologie de matériel implanté.

L'architecture électrique est bâtie autour d'un ordinateur industriel, muni d'un Pentium III d'Intel MMX 600MHz comme processeur central. C'est lui qui servira de centre de décisions au robot. Le besoin en mémoire dépend de la fréquence de données entrantes, le temps d'acquisition, et le nombre de variables d'entrées. De plus, si l'interface graphique s'exécute sur un autre poste ou station, l'ordinateur exécutant le contrôle pourrait avoir besoin moins de 1 méga byte de RAM. Pour faciliter le développement, nous pouvons le relier, grâce à une interface Réseau-Ethernet, à un réseau local et même à Internet ce qui permet le développement à distance. À cet ordinateur central sont attachés plusieurs périphériques, notamment, une carte d'entrée/sortie, modèle PCISCC5P, multi-fonctionnelle générant des signaux PWM pour les cinq canaux et une autre carte de type S626, pour lire les signaux des encodeurs et envoyer des sorties digitales pour l'arrêt et le sens de rotation du moteur. Ces deux cartes multi-fonctionelles qui permettent la communication avec plusieurs périphériques, donnent accès, entre autre aux détecteurs de position et de l'envoi du signal PWM aux cartes de puissance des moteurs du robot. Expliquons brièvement la fonctionnalité de chacun de ces périphériques.

4.4.1 Alimentation

Le robot peut s'alimenter d'une source de courant usuelle, qui est un bloc d'alimentation externe. Ensuite, l'alimentation est amenée vers un convertisseur qui fournit des sources stables de 5 volts, 12 volts et –12 volts qui sont utilisées pour alimenter l'ordinateur et tous ses périphériques. Nous avons aussi une source de 24V qui est utilisée pour alimenter tous les moteurs et qui peut être interrompue par un bouton d'urgence. L'alimentation électrique est fournie par une armoire indépendante de l'ordinateur ; par contre la commande est transmise à l'intermédiaire d'un câble par lequel passent aussi toutes les informations de position et de vitesse de déplacement.

4.4.2 L'ordinateur de contrôle

Le contrôleur est le composant électrique principal du système. Toutes les informations venant et allant vers les différents actionneurs du robot, sont traitées par le contrôleur qui calcule en temps réel les ordres de commande.

Le logiciel de programmation est enregistré dans la mémoire centrale de l'ordinateur. Ce logiciel interprète les instructions de commande, et le contrôleur transmet ces instructions de la mémoire centrale vers les différents actionneurs du robot.

Grâce aux encodeurs incrémentaux et la carte S626, le contrôleur reçoit des informations de position pour chacun des axes. Ceci permet un contrôle en boucle fermée des mouvements du robot.

4.4.3 Carte d'entrée/sortie PCIDCC5P pour le signal PWM

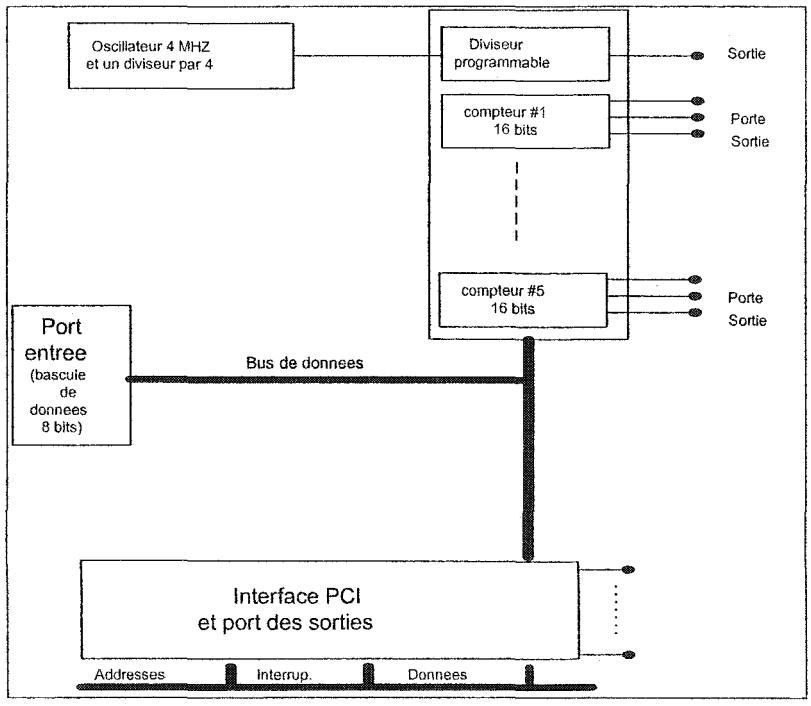


Figure 18 Diagramme bloc de la carte PCI DCC5P

La figure 18 illustre le bloc fonctionnel de la carte à cinq compteurs PCIDCC5-P. Cette carte fait partie de la série des cartes de compteurs/temporisateurs qui contenaient le fameux circuit du système de contrôle du temps LSI noté AM9513. Ce circuit AM9513 comprend cinq compteurs 16 bits indépendants de niveaux haut/bas. La présente carte est dotée d'un seul circuit AM9513, et par conséquent, cinq compteurs ; mais on pourrait avoir 20 compteurs en disposant de la carte PCIDCC20-P. En tout cas, on a une entrée de 8 bits et une sortie de 8 bits en plus du circuit AM9513. Les compteurs peuvent être programmés pour compter les niveaux haut ou bas en binaire ou BCD. Une bibliothèque de procédures, « dcc_pwm_out_lib.h » et « libOpalDcc10p.a », sont installées avec les pilotes pour mieux exploiter efficacement la carte.
Le tableau VI montre les principales fonctions utilisées.

Tableau VI

Les principales routines de la carte PCIDCC5-P

Nom de la routine	Description
InitDccCard()	Initialiser la carte.
SetDccCardFrequencyAndDutyCycle(carte,Nc,x,f,rc)	Envoyer un signal PWM de rapport cyclique rc entre 0 et 1, à l'axe x, avec une fréquence f.
CloseDccCard()	Fermer la carte.

4.4.4 Les capteurs encodeurs

Tous les contrôles de mouvements mécaniques nécessitent un encodeur angulaire effectuant le lien entre l'entraînement et la commande. Les encodeurs angulaires transforment les différents mouvements de rotation en signaux électriques. Le cœur de

l'encodeur optique est formé d'un disque, disposant d'un nombre variable de segment opaque et transparent, détecté par une source lumineuse. La structure définit la résolution ainsi que la précision de positionnement du mouvement à contrôler.

Deux systèmes de mesure séparés optiquement génèrent deux séquences d'impulsions déphasées électriquement de 90°. Une reconnaissance du sens de rotation est ainsi rendue possible.

Les encodeurs émettent des signaux de sortie digitaux. Une période de signal peut être divisée en 4 phases de mesure lorsqu'on analyse l'écart des deux fronts CHA et CHB (Figure19) on obtient de cette manière une quadruple exploitation du nombre d'impulsion utilisé ce qui augmente la résolution de la capture de position.

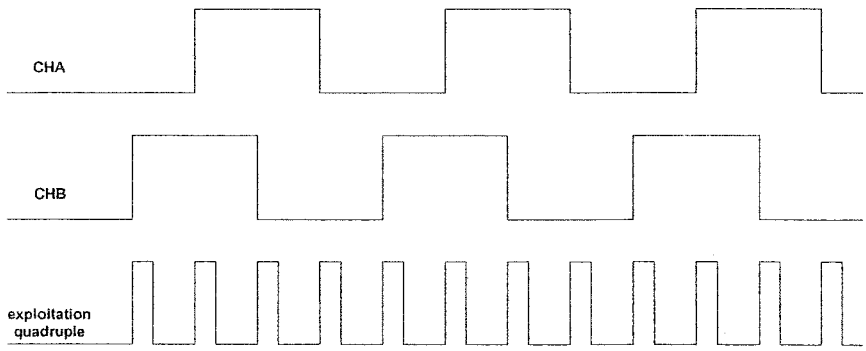


Figure 19 Exploitation quadruple

D'autre part, la détection de position se fait par comptage ou décomptage des fronts montants et des fronts descendants provenant du canal CHA et CHB.

De plus, les détecteurs de position sont placés sur les arbres des moteurs et nous permettent de localiser la position articulaire de chaque axe par rapport à sa position initiale en exploitant les impulsions venant des cinq encodeurs réglés de façon à donner une meilleure précision.

Le capteur est en mesure d'accomplir plusieurs tâches:

- Mise à jour de la position et de l'orientation du robot ;

- Extraction des informations nécessaires à la paramétrisation des trajectoires.

4.4.5 Carte multi-fonctionnelle S626

Le modèle S626 est une carte d'interface du système contrôlé, elle fournit 3 horloges, 8 A/D, 8 D/A, 8 E/S digitales. L'une des horloges est utilisée pour générer l'interruption de contrôle du temps. La figure 20 montre le diagramme bloc de la carte de Sensoray modèle S626. Elle dispose de six compteurs 24-bit souples pour des encodeurs à quadrature, d'une génération d'interruption , d'une protection de batterie des compteurs, de quatre sorties analogiques de 13-bit, qui peuvent être programmées pour +/- 10 volts. Un circuit unique empêche les sorties imprévisibles de D/A pendant l'initialisation d'unité centrale de traitement. Aussi, la carte est dotée de 16 entrées différentielles analogiques/numériques avec une résolution 14-bit, et de 48 canaux numériques bi-directionnels.

À la différence des compteurs conventionnels, les compteurs du 626 n'accumulent pas des erreurs : quand l'encodeur hésite ou change la direction, la carte S626 l'approvisionne en tension de 5 volts à chaque canal d'encodeur. Chaque compteur peut être programmé pour fonctionner comme temporisateur à l'aide de l'horloge interne du 626 ou d'une horloge externe. Une fois appareillés, les compteurs 48-bit peuvent être employés pour la mesure de fréquence.

D'autre part, l'interface à prix réduit du modèle 626 de Sensoray ménage de l'espace en combinant cinq fonctions dans un module de demi-taille. Cinq câbles plats relient le modèle 626 aux périphériques externes.

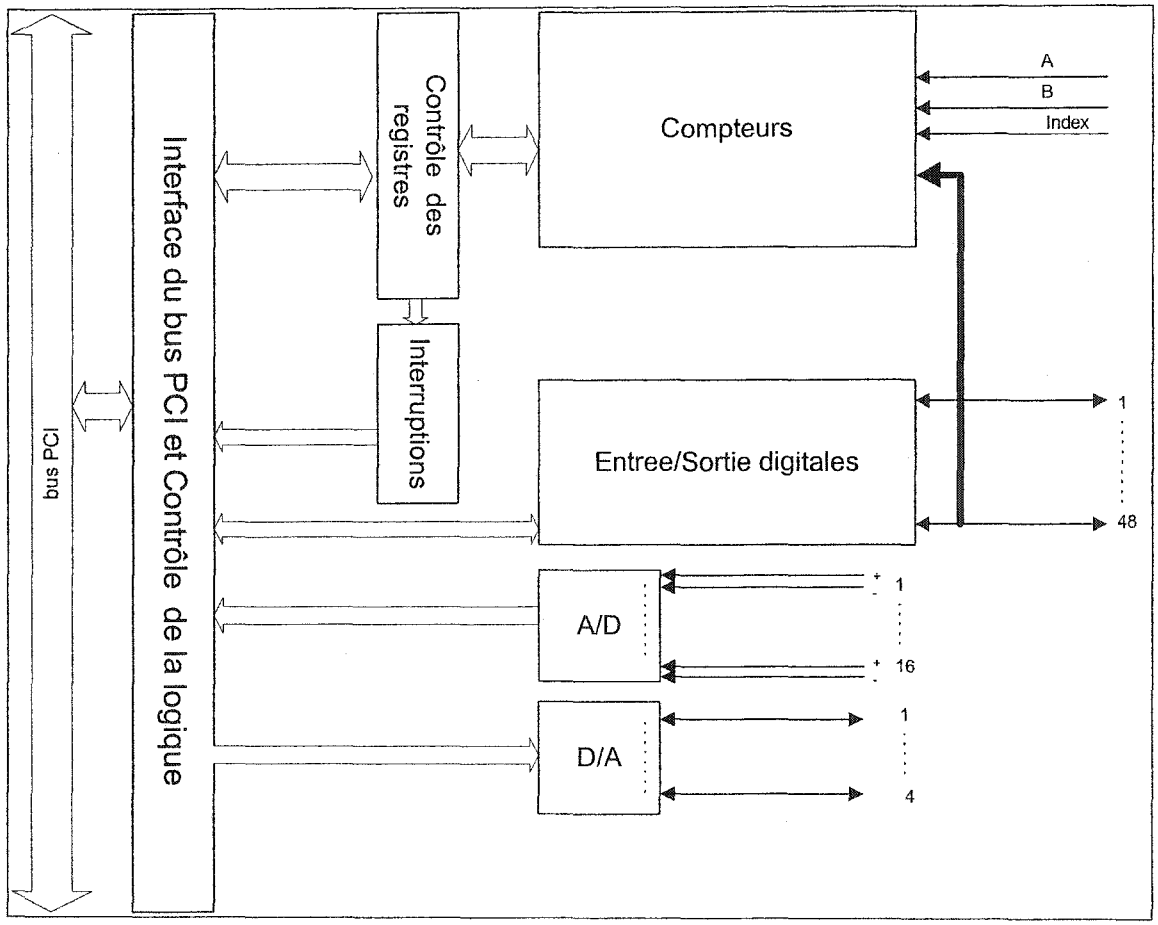


Figure 20 Diagramme bloc de la carte Sensoray 626

De plus, l'utilisation de cette carte nécessite une librairie de code « lib_enc_s626.h » pour le développement. Tous les dispositifs du modèle 626 peuvent être exploité en utilisant le kit de développement de logiciel, « S626_enc_lib.h » et « libOpalSensoray626.a ». Ce DLL de 32 bits pour QNX permet une utilisation aisée de la carte 626. Ainsi le tableau VII illustre les routines les plus utiles de cette librairie.

Tableau VII

Les principales routines de la carte S626

Nom de la routine	Description
S626_init()	Initialiser la carte.
ResetEnc(carte,x)	Mise à zéro du compteur d'impulsions de l'axe x.
ReadEnc(carte,x)	Lecture d'impulsions de l'encodeur x.
S626_Close()	Fermeture de la carte.

4.5 Design informatique

Les anciens travaux faits sur le robot MAKER 100, étaient dotés du système d'exploitation DOS. Cependant, ce système nous posait d'importantes contraintes quant à la programmation en temps réel. Pour pallier à ces inconvénients majeurs, nous avons choisi de changer le système d'exploitation pour le développement. Nous avions le choix entre Windows CE, QNX, VxWorks et Linux RT. Windows est très populaire, mais complexe à l'intégration de périphériques. Les trois autres nous donnaient tous des avantages équivalents quant à la programmation temps-réel, la gestion des processus et l'intégration de nouveaux périphériques. Notre choix s'est cependant porté sur QNX pour deux raisons principales. La première est la gratuité de ce système d'exploitation pour des fins non commerciales et l'accès facile à des mises à jour. En second lieu, la disponibilité des personnes ressources (via le web) nous a grandement servi lors du développement de nos procédures.

Le programme est divisé en 4 phases. La première phase contenait les outils nécessaires pour le développement d'une interface graphique interactive sur QNX. La deuxième phase comprend les modules de la simulation du contrôleur numérique à implanter. En troisième phase, on réalise une communication avec les différents périphériques, suivie d'une identification des moteurs. Et la dernière regroupe les trois précédentes pour une

implantation en temps réel. Depuis son exécution, le programme utilisé a réussi largement tous les tests d'exécution.

4.6 Programmation et résultats

4.6.1 Simulation

Le manipulateur modélisé dans le cadre de ce travail, est un seul robot MAKER 100 à 5 ddl détaillé dans le chapitre 2. Ce robot est soumis à une loi de commande adaptative selon la théorie de Slotine et Li comme décrite au chapitre 3.

Avant de se pencher sur le temps réel, il faut s'assurer d'une bonne performance du contrôleur mis en place. Pour ce faire, on commence par une simulation. La figure 21 montre l'organisation des modules composant le programme de simulation du contrôleur appliqué au robot.

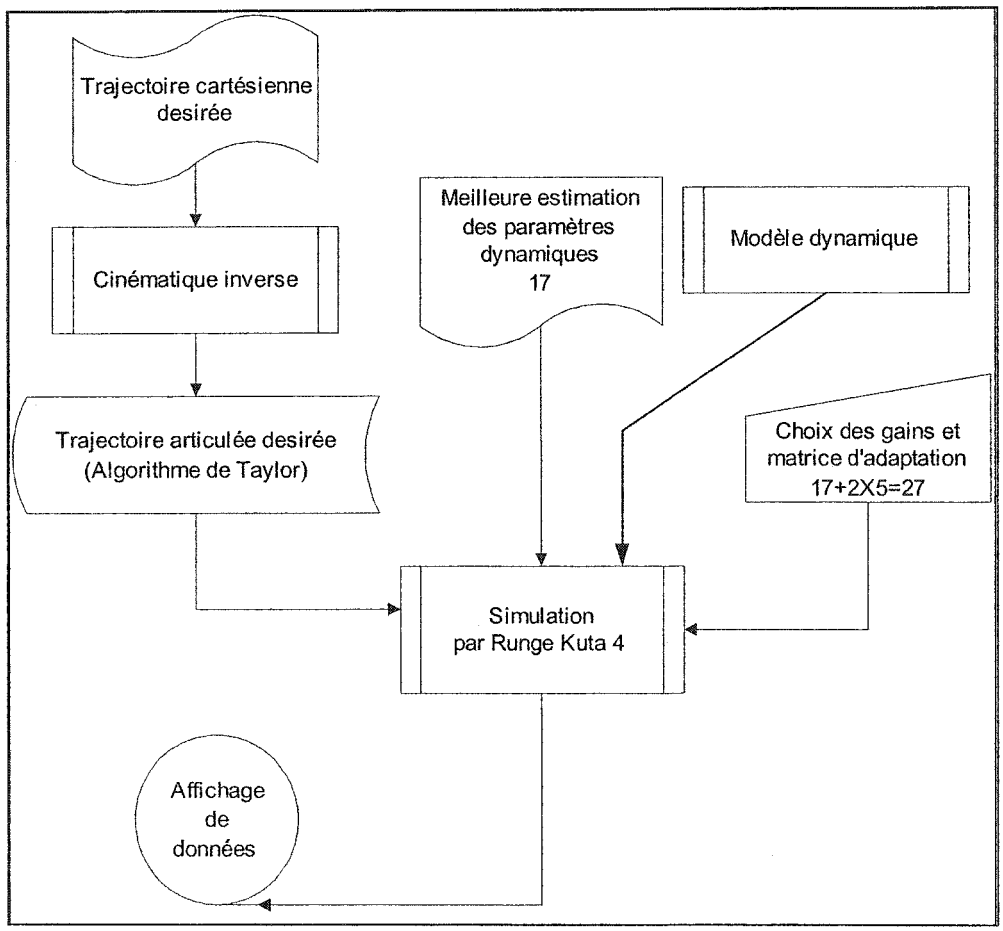


Figure 21 Organigramme de simulation hors-ligne du contrôleur

Et afin de valider tout résultat de cette simulation, nous avons procédé au développement de trois versions de programmes : le premier est avec l'aide de Simulink, le deuxième est avec le script de Matlab, et le troisième, qui nous intéresse, en langage C standard sur la plate-forme QNX. En plus, nous faisons appel à une interface graphique développée localement à cette fin, grâce à l'outil de développement Builder de Neutrino-QNX semblable à Visual Basic de Microsoft pour Windows. Elle permet un affichage des données sous des formats aisément accessibles. Cette interface permet aussi à l'utilisateur de choisir les sommets de la trajectoire désirée sous forme d'une matrice qui contient des valeurs par défaut valides, Il peut aussi fixer la durée du

parcours de la trajectoire, ainsi que la tolérance de déviation bornée permise. La figure 22 montre la fenêtre principale du programme.

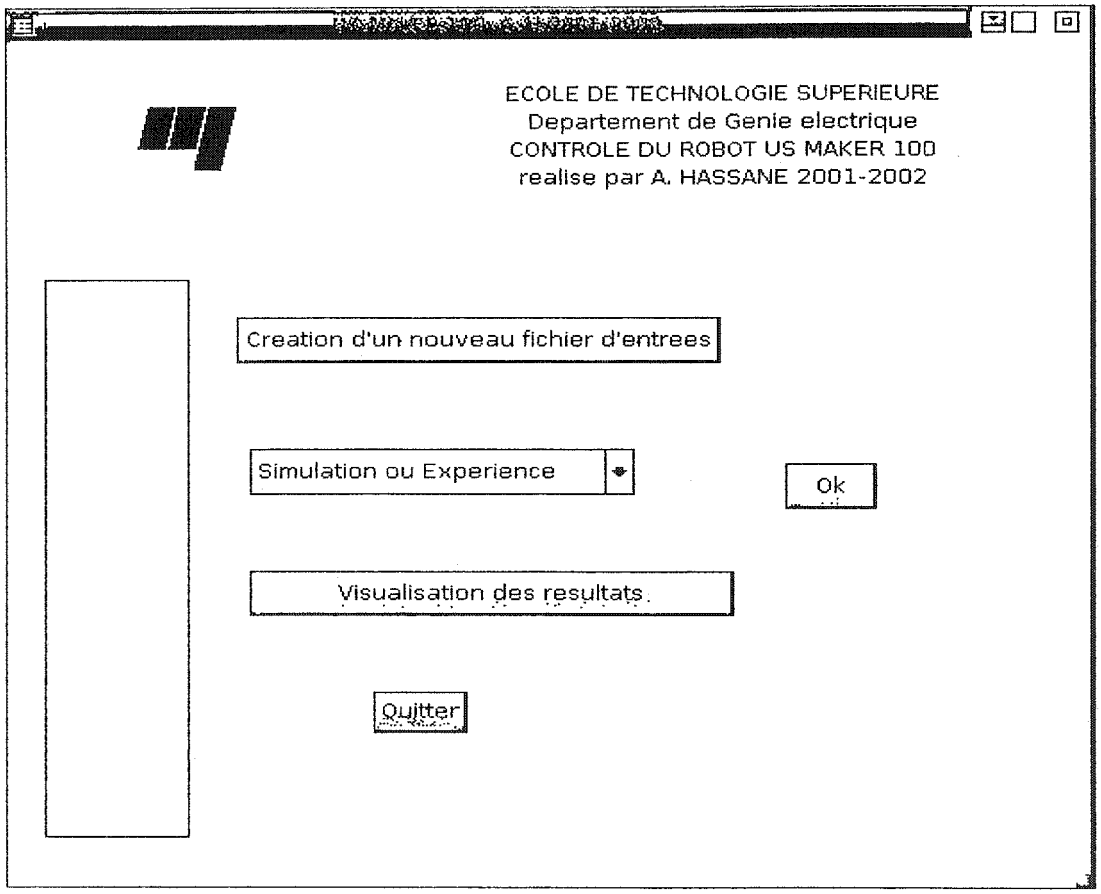


Figure 22 Menu principal du programme

De plus, l'utilisateur définit les gains de contrôle, la matrice d'adaptation et sa meilleure estimation des paramètres dynamiques.

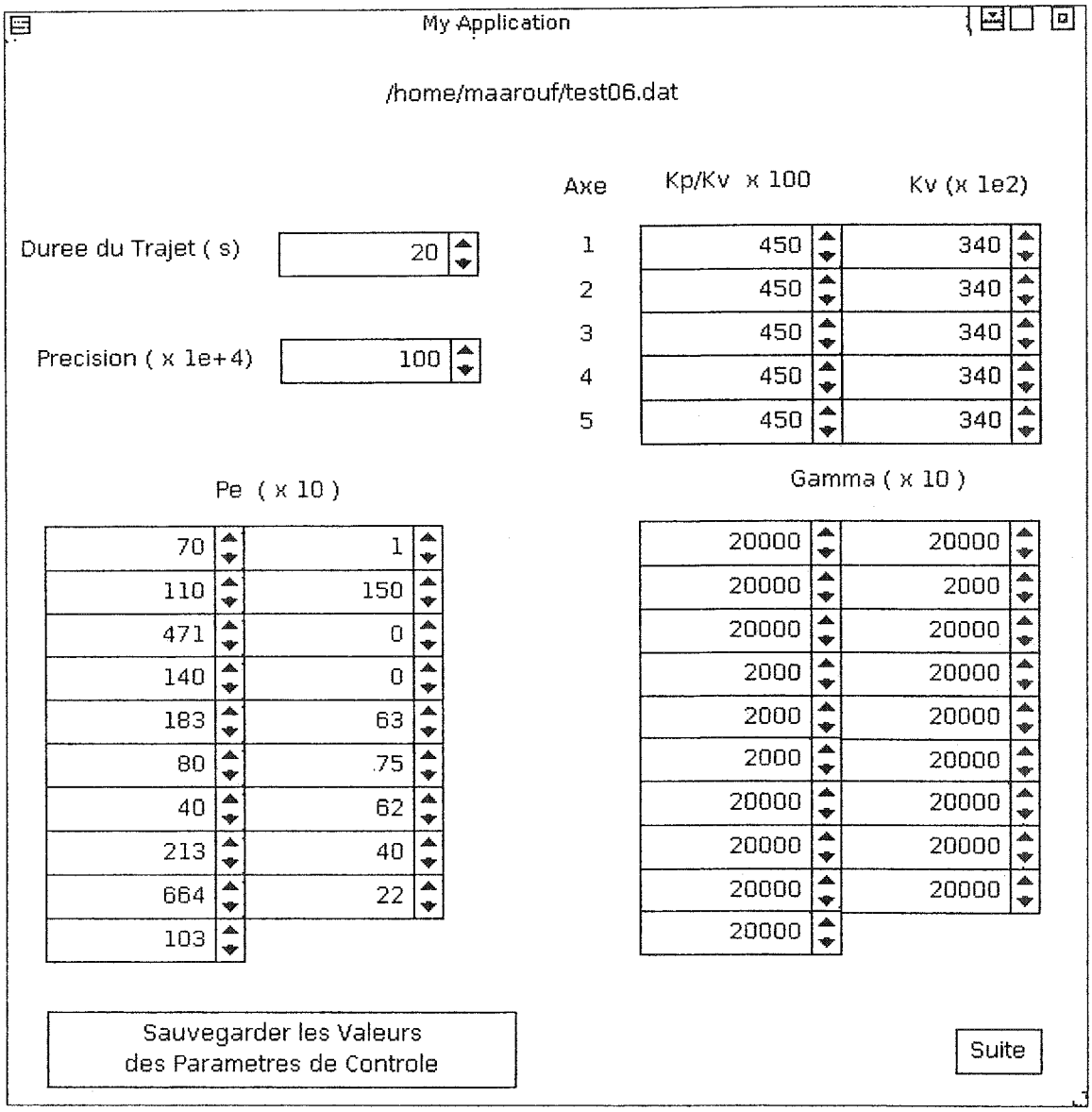


Figure 23 Menu du choix des paramètres de contrôle

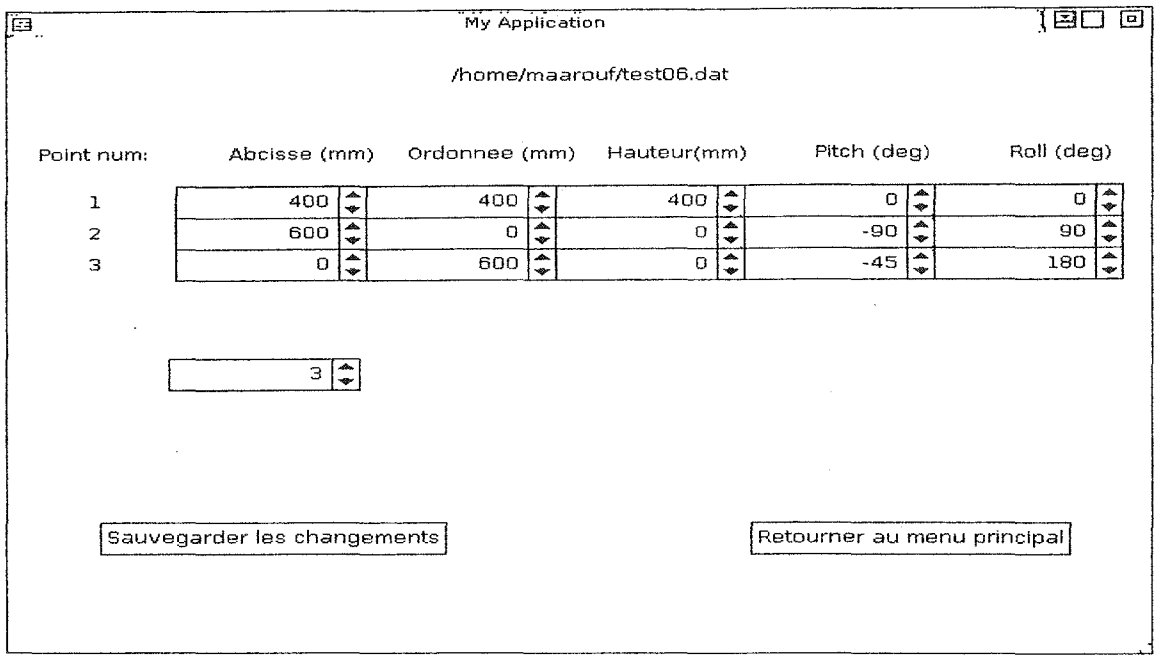


Figure 24 Fenêtre de spécification des points du triangle.

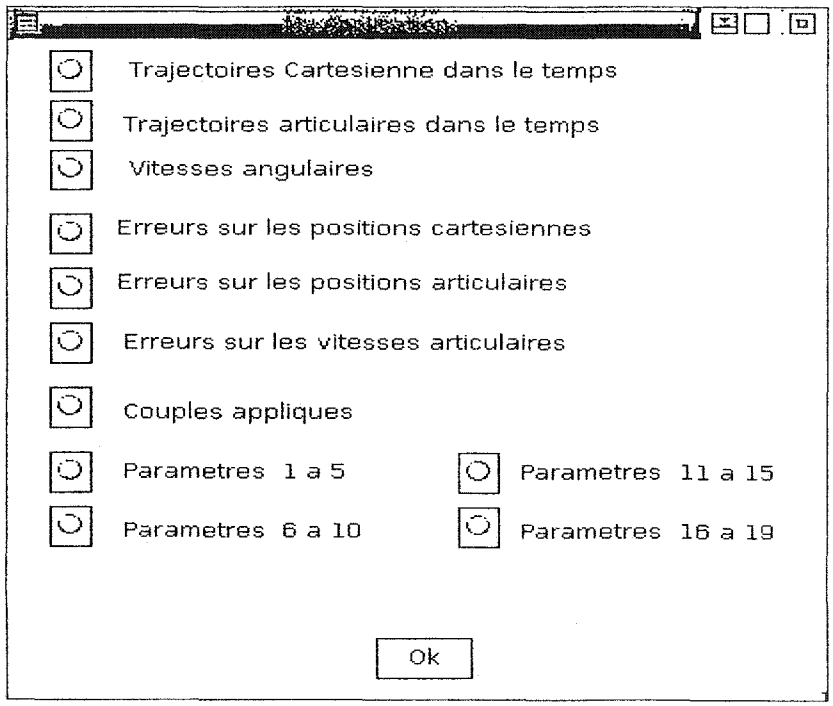


Figure 25 Menu pour la visualisation des graphes

Les figures 22 à 25 illustrent un exemple d'utilisation de l'interface graphique.

Ensuite, l'utilisateur pourrait lancer la simulation pour visualiser les trajectoires cartésiennes et articulaires, les courbes de vitesses, les erreurs entre l'allure désirée et mesurée pour la position et la vitesse, l'évolution des paramètres dynamiques estimés ;

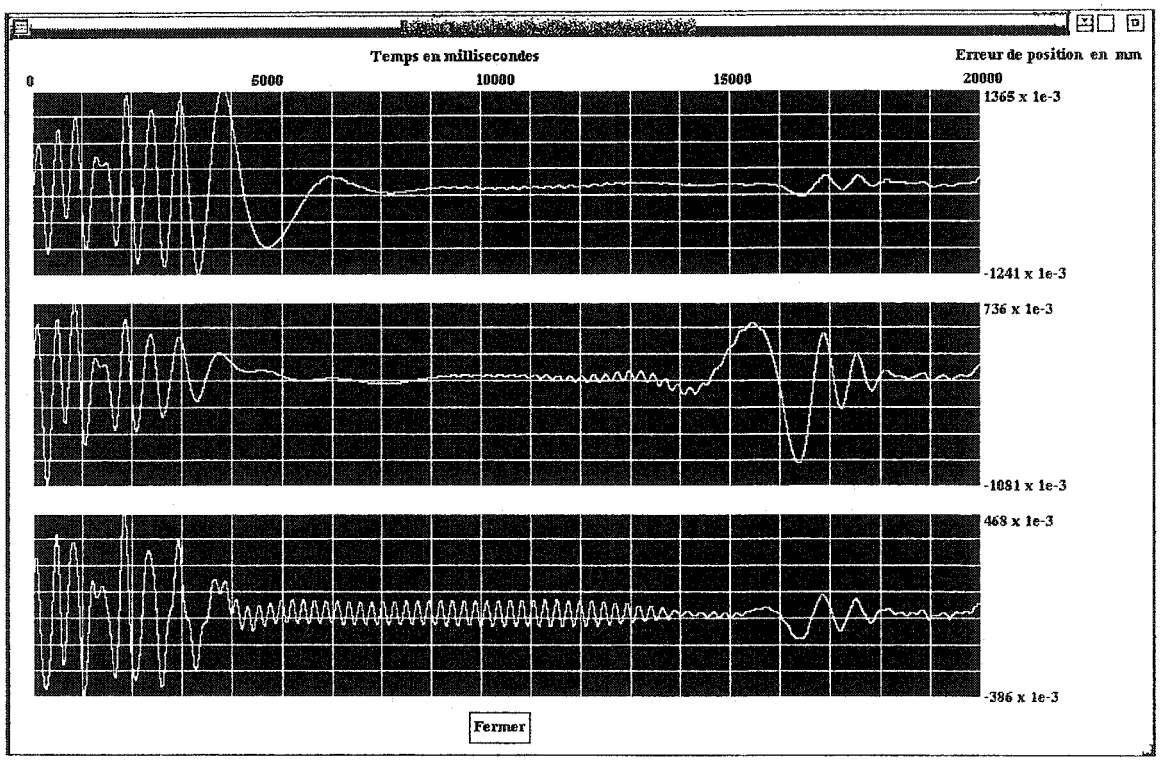


Figure 26 Graphe des erreurs de position cartésienne

La figure 26 montre les erreurs de position cartésienne de la simulation. De plus, la figure 27 présente les erreurs de vitesses dans l'espace articulaires. De même, la figure 28 illustre la variation des paramètres dynamiques estimés en fonction du temps. Et la figure 29 affiche les couples appliqués aux articulations.

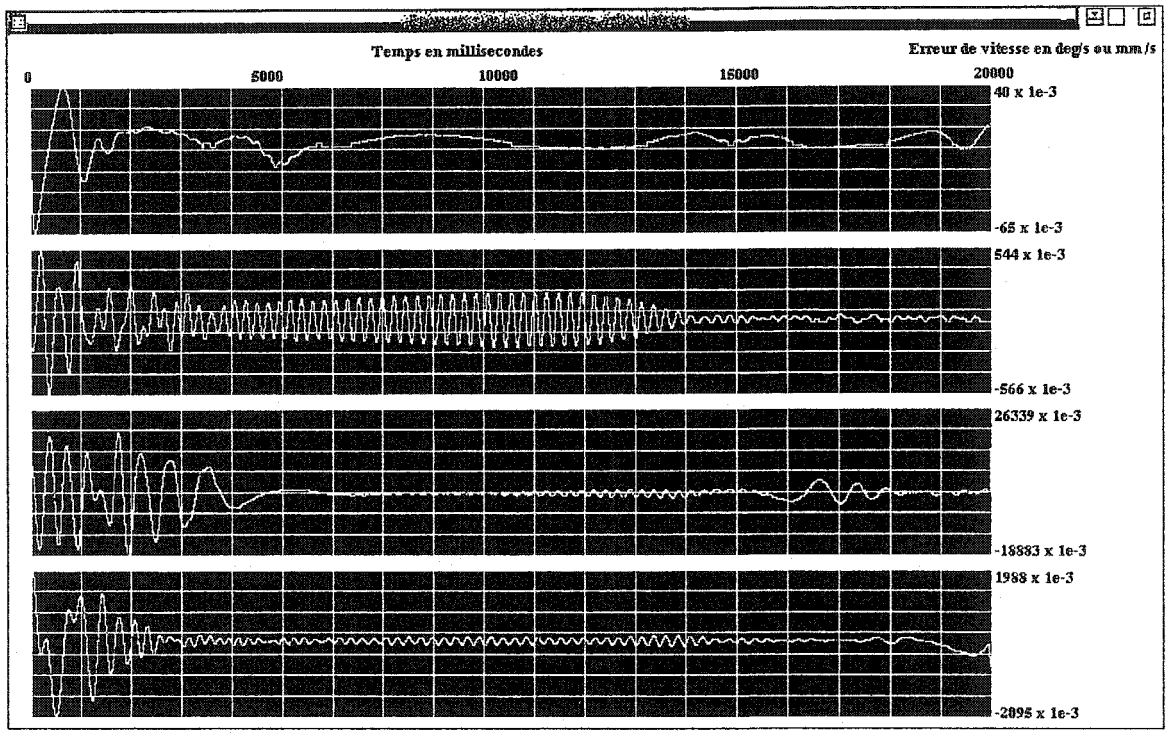


Figure 27 Graphe des erreurs de vitesses

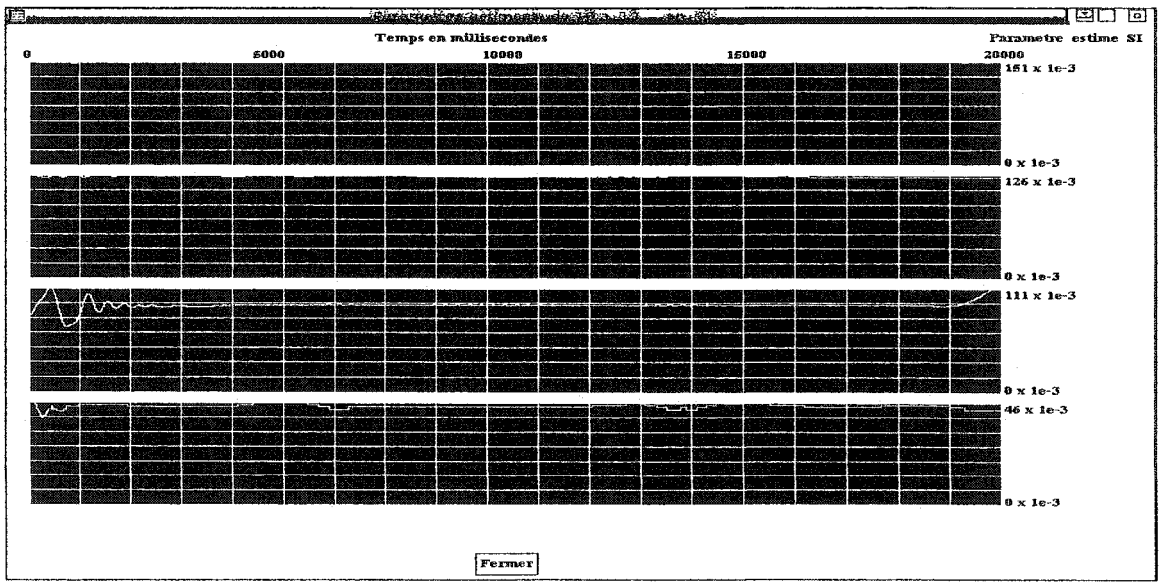


Figure 28 Graphe de l'évolution des paramètres dynamiques estimés

Et aussi les couples appliqués aux différentes articulations.

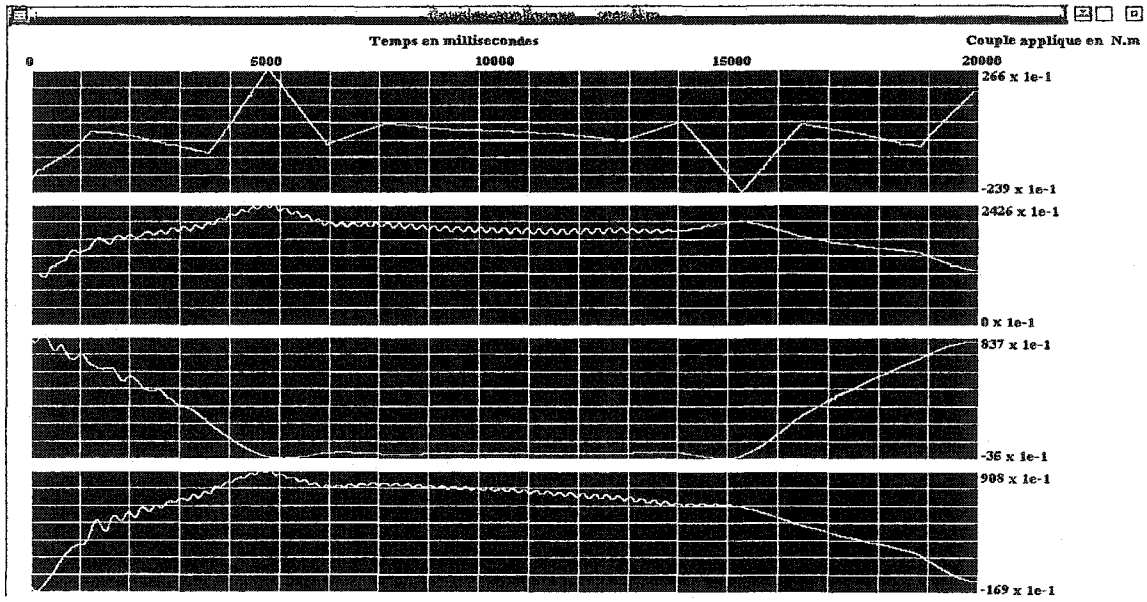


Figure 29 Graphe des couples appliqués aux articulations

Ces résultats sont basés sur les valeurs numériques suivantes pour les paramètres dynamiques :

Tableau VIII

Valeurs numériques des paramètres dynamiques

p_1	7	p_6	8	p_{11}	1	p_{15}	6.3
p_2	11	p_7	4	p_{12}	15	p_{16}	7.5
p_3	47.1	p_8	21.3			p_{17}	6.2
p_4	14	p_9	66.4			p_{18}	4
p_5	18.3	p_{10}	20.3			p_{19}	2.2

Ces paramètres sont calculés à partir des données suivantes :

Tableau IX

Données numériques de la simulation

	Matrices d'inertie			Masse du lien i	Centres de masse
Articulation	Ixx	Iyy	Izz	m_i	z_i ou y_i
1	3	4	7	8.71	1.5
2	5	4	3	7.5	2
3	3	4	7	9.15	1
4	5	3	4	3.15	2
5	2	2	1	1.7	1

4.6.2 Contrôle en temps réel

4.6.2.1 Etalonnage géométrique

L'étalonnage géométrique des robots consiste à identifier les valeurs des paramètres des modèles géométriques. Il permet donc d'améliorer la précision de positionnement de l'organe terminal du robot. Ces paramètres sont les distances et angles entre les axes de repères fixés aux corps, les « offsets » articulaires, la flexion due à l'action de la gravité,...etc.

La procédure classique d'étalonnage consiste à estimer les paramètres géométriques par minimisation de la norme de l'écart entre les situations de l'organe terminal mesurées par un capteur externe et les valeurs prédites par le modèle géométrique direct.

Et ensuite, on fait correspondre ces valeurs avec le nombre d'impulsions expédiées par les encodeurs angulaires placés sur les arbres des moteurs. Les résultats suivants, résumés dans le tableau X, ont été obtenus :

Tableau X

Correspondance des impulsions avec les données géométriques

Axe	Limites mécaniques (degrés ou mm)	Limites logicielles (degrés ou mm)	Impulsions (unités/degrés ou mm)
1	Entre 0 et 350.452 deg	Entre 0 et 348 deg	883 /deg
2	Entre –141 et 141 deg	Entre –140.4 et 140.4 deg	497 / deg
3	Entre 400 et 910 mm	Entre 400 et 901 mm	96 / mm.
4	Entre –114 et 114 deg	Entre –110.45 et 110.45 deg	38 / deg
5	Entre 0 et 348.49 deg	Entre 0 et 348.49 deg	489 / deg

4.6.2.2 Identification des moteurs

Le montage proposé ici permet de commander un moteur à courant continu en contrôlant sa vitesse et son sens de rotation via la carte PCIDCC5-P et la carte amplificatrice de puissance. Le montage repose sur le principe du hachage de la tension d'alimentation. la figure 30 explique comment en faisant varier le rapport cyclique de la tension d'alimentation on peut contrôler la puissance électrique fournie à un moteur (et donc sa vitesse).

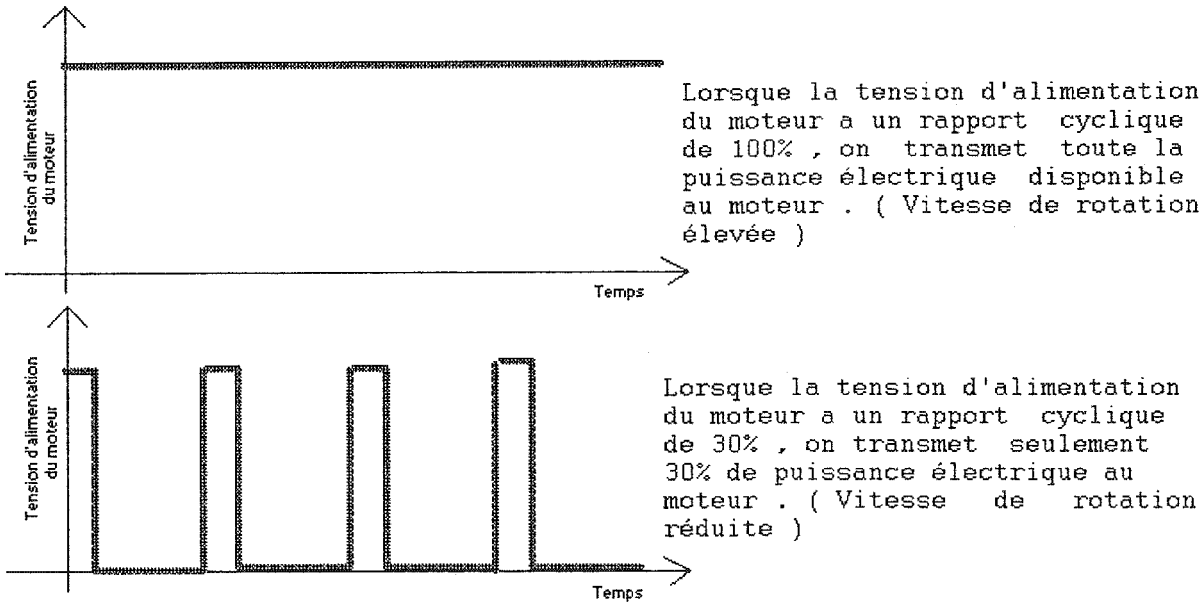


Figure 30 Signal PWM à un rapport cyclique variable pour un axe du moteur

Ces travaux expérimentaux ont permis de valider notre approche, basée sur des techniques simples d'analyse de modèle, de filtrage et de moindres carrés. En particulier, moyennant un bon traitement des données (filtrage), la méthode ne présente pas de biais significatif.

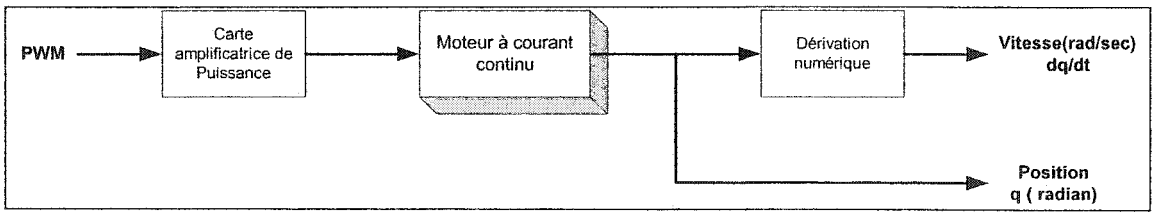


Figure 31 Montage en boucle ouverte.

A titre d'exemple, on fait varier le signal PWM entre les valeurs permises et on mesure par une méthode numérique la vitesse statique de l'axe 5 du robot MAKER 100, à partir de la position lue par l'encodeur, puis on trace cette variation représentée par la figure 32 pour identifier les paramètres du moteur.

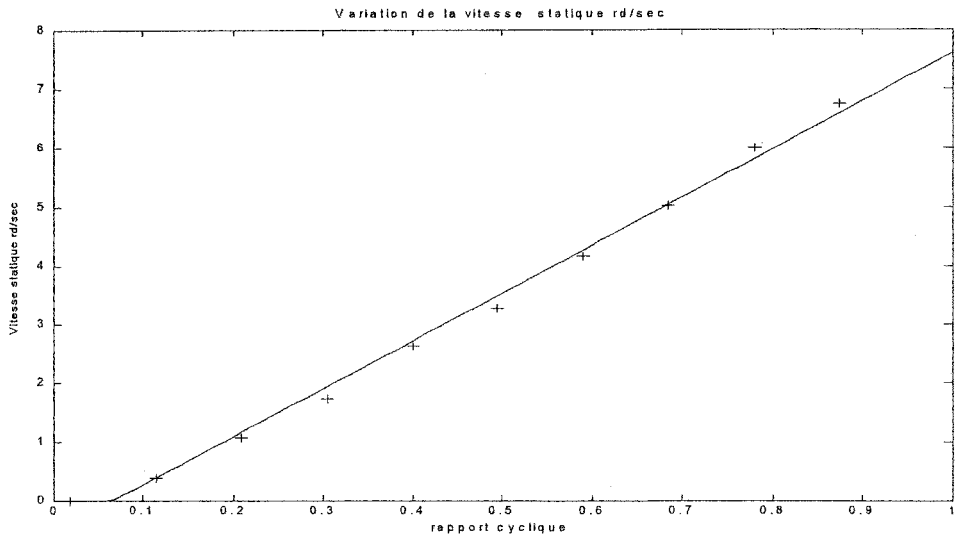


Figure 32 Variation de la vitesse statique de l'axe 5 en fonction du PWM

Ensuite on fixe le rapport cyclique du signal PWM à 0.5, et on suit l'évolution de la position et de la vitesse comme le montre la figure 33.

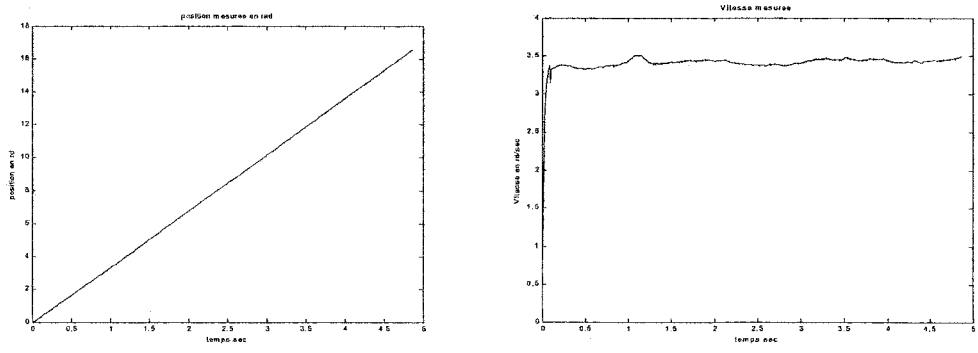


Figure 33 Évolution de la position et la vitesse de l'axe 5

4.6.2.3 Conversion du couple en un signal PWM

Le principe de base de la conversion d'un couple appliqué aux bornes d'un moteur à courant continu en un signal PWM (modulation de largeur d'impulsions), est basé sur la théorie des machines électriques, qui est déduite des circuits électromagnétiques de la machine tournante. Donc, si on applique une tension continue va aux bornes du moteur, celui-ci tourne avec une vitesse Ω et on aura les équations suivantes en régime statique :

Force électromotrice : $E_a = k.\Phi.\ \Omega$

Equation de la maille : $va = E_a + R_a.I_a$

Puissance du moteur: $P_m = E_a.\ I_a = \tau.\Omega \Rightarrow \tau = k.\ \Phi.\ I_a$

En combinant ces équations on aura:

$$v_a = k\Phi.\ \Omega + R_a.\ \tau / k\Phi. \qquad (4.1)$$

avec :

R_a : résistance des armatures du moteur.

I_a : courant absorbé par l'induit.

Φ : flux magnétique à travers les spires.

k : constante du moteur.

De plus, le signal de commande PWM est un signal carré à une fréquence constante (20 kHz dans notre cas), et une amplitude Van qui est la tension nominale du moteur, mais à un rapport cyclique variable *rc*. Alors que la tension continue équivalente à ce signal vaut :

$$v_a = V_{an}.rc. \qquad (4.2)$$

En combinant les équations 4.1 et 4.2 on obtient l'équation 4.3.

$$rc = (k\Phi.\ \Omega + R_a.\ \tau / k\Phi)/V_{an}. \qquad (4.3)$$

4.6.2.4 Implantation du contrôleur PD

4.6.2.4.1 Principe de contrôle

L'idée fondamentale d'une boucle d'asservissement est de prendre la commande désirée de position, et d'envoyer cette commande au moteur à l'aide des cartes amplificatrices de puissance, ce qui fait tourner le moteur, puis de mesurer cette position via les encodeurs et la comparer à la position désirée (fig. 34). La différence s'appelle le signal d'erreur identifié « e ». Le type de contrôleur détermine quel calcul le système de commande exécute sur le signal d'erreur. Un type de contrôleurs est le contrôleur PD dont la loi de commande est exprimée par la relation suivante:

$$e(t) = q_d - q.$$
$$\tau = k_p.e(t) + k_d.\dot{e}(t)$$

L'objectif spécifié est une consigne de position et par conséquent se déduit la consigne de vitesse. L'application de commande repose sur la phase de modélisation à partir d'essais spécifiques de la partie précédente. Le robot devant fonctionner avec ses propres charges. Dans le cas présent, les actions sont calculées sous la forme d'une combinaison d'un échelon, d'une rampe, d'une parabole et d'une cubique. La période d'échantillonnage est de 4 millisecondes et le temps de réponse fixé en boucle fermée est de 5 s. La précision obtenue sur l'écart de poursuite est inférieure à 0.025 radians pendant le déplacement et est pratiquement nulle en statique.

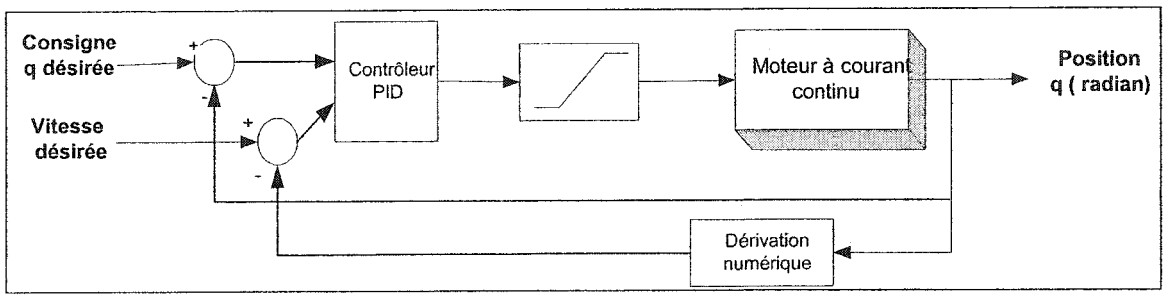


Figure 34 Montage en boucle fermée

Le travail effectué durant cette phase du projet, était de déterminer des paramètres de contrôle qui conduisent à une meilleure performance lors de la réalisation d'une tâche par le manipulateur. La méthode consiste en ajustement manuel des gains à partir des valeurs préalablement choisies, jusqu'à une satisfaction de la performance obtenue.

Le tableau XI résume les paramètres retenus pour le contrôle du robot.

Tableau XI

Paramètres de contrôle pour le robot MAKER 100

Axe	PWM minimale (%)	Gain dérivatif Kd	Gain proportionnel Kp
1	16	0.2	136
2	8	0.2	155
3	6	0.2	6.75
4	4	0.2	8
5	6	0.0	270

4.6.2.4.2 Résultats expérimentaux

L'expérience faite sur le robot MAKER 100 en temps réel est de prendre comme consigne la trajectoire générée par l'algorithme de Taylor à partir du triangle décrit au tableau IV et en prenant l_5=0. La figure 35 montre, dans l'espace cartésien, les deux

trajectoires cartésiennes désirée et réelle de l'effecteur du robot pendant un laps de temps de 20 secondes.

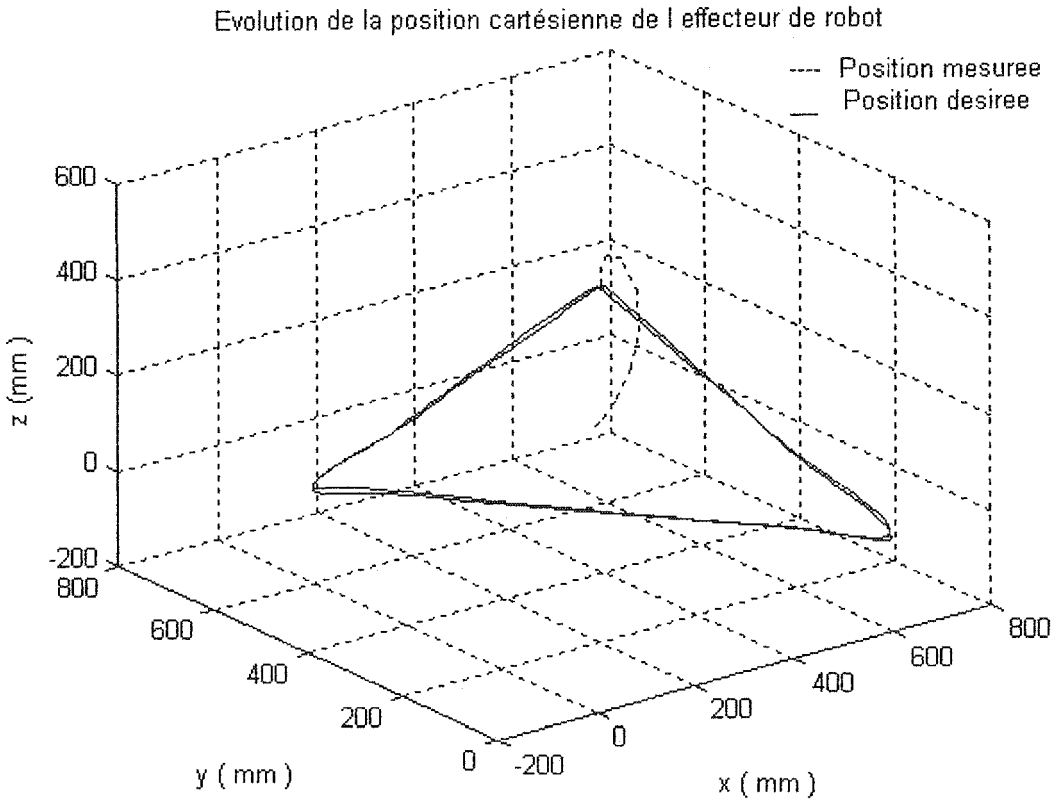


Figure 35 Parcours dans l'espace cartésien de l'effecteur du robot

Ensuite, pour négliger l'effet de la pesanteur, nous avons pris comme consigne 2 un segment horizontal dont les sommets sont donnés au tableau XII.

Tableau XII

Coordonnées de l'effecteur du robot aux sommets d'un triangle (consigne 2)

	X (m)	Y (m)	Z (m)	roulis (rd)	tangage (rd)
Point A	0.604	0.	0.	0	0
Point B	0.6	0	0	$-\pi/2$	$+\pi/2$
Point C	0	0.6	0	$-\pi/4$	π

Et la figure 36 illustre, dans l'espace cartésien, les deux trajectoires désirée (consigne 2) et réelle effectuée par le robot pendant une durée de 20 secondes.

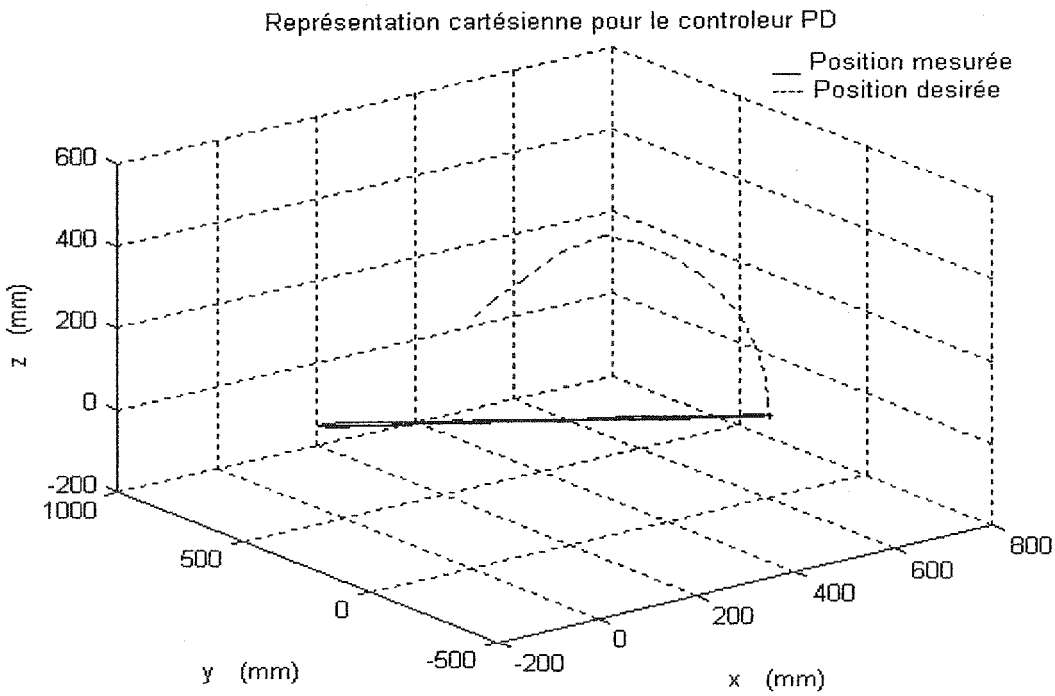


Figure 36 Parcours dans l'espace cartésien de l'effecteur du robot

Ensuite, la figure 37 montre la poursuite de trajectoires dans le domaine articulaire.

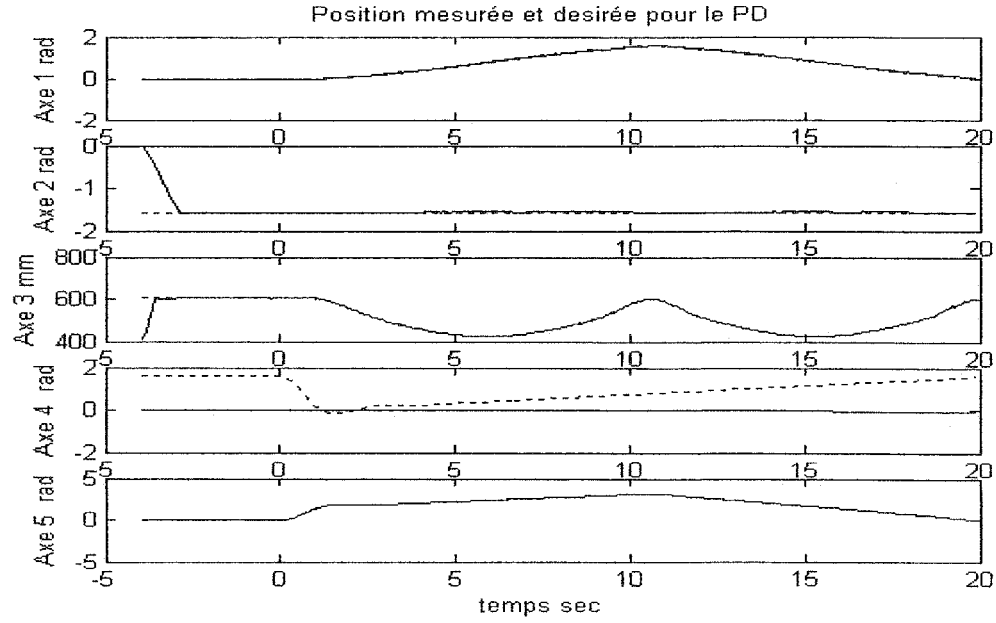


Figure 37 Évolution des articulations désirées et réelles pendant le mouvement

De plus, la figure 38 présente l'évolution du signal de commande PWM envoyé aux cartes de puissance des cinq articulations.

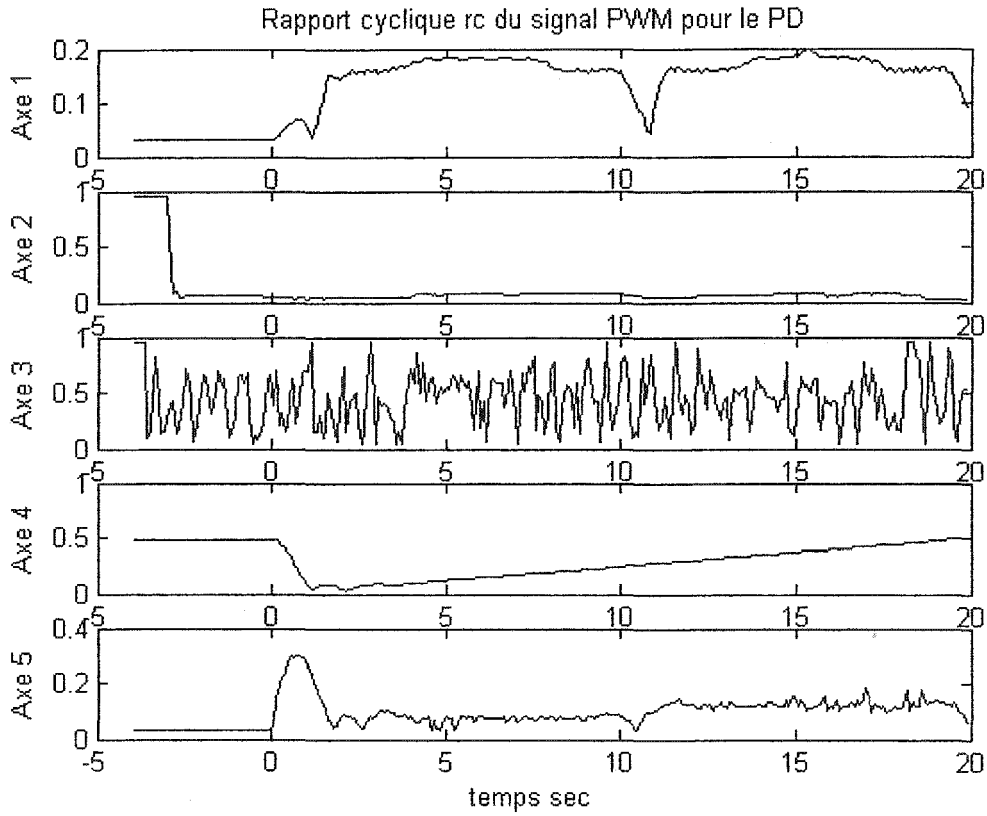


Figure 38 Les commandes appliquées aux articulations pendant le mouvement

La figure 39 illustre l'erreur de poursuite dans l'espace articulaire des cinq articulations. Mais, la figure 40 montre l'erreur de poursuite dans le domaine cartésien.

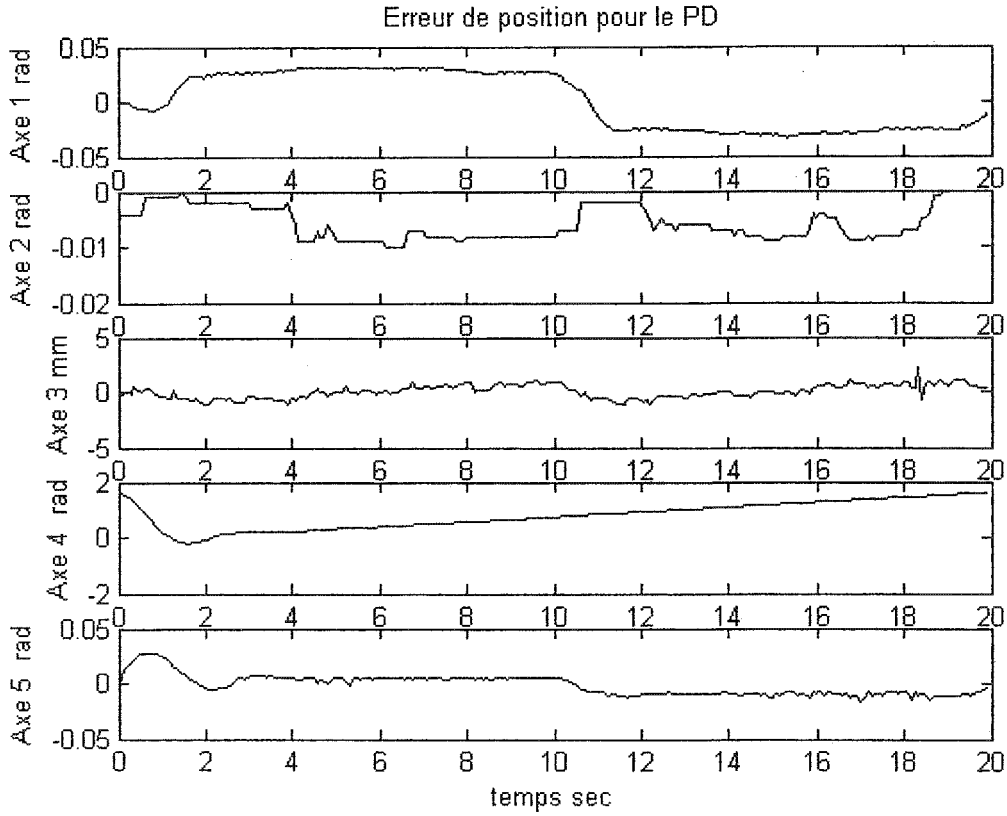


Figure 39 Les erreurs sur les positions articulaires du robot en mouvement

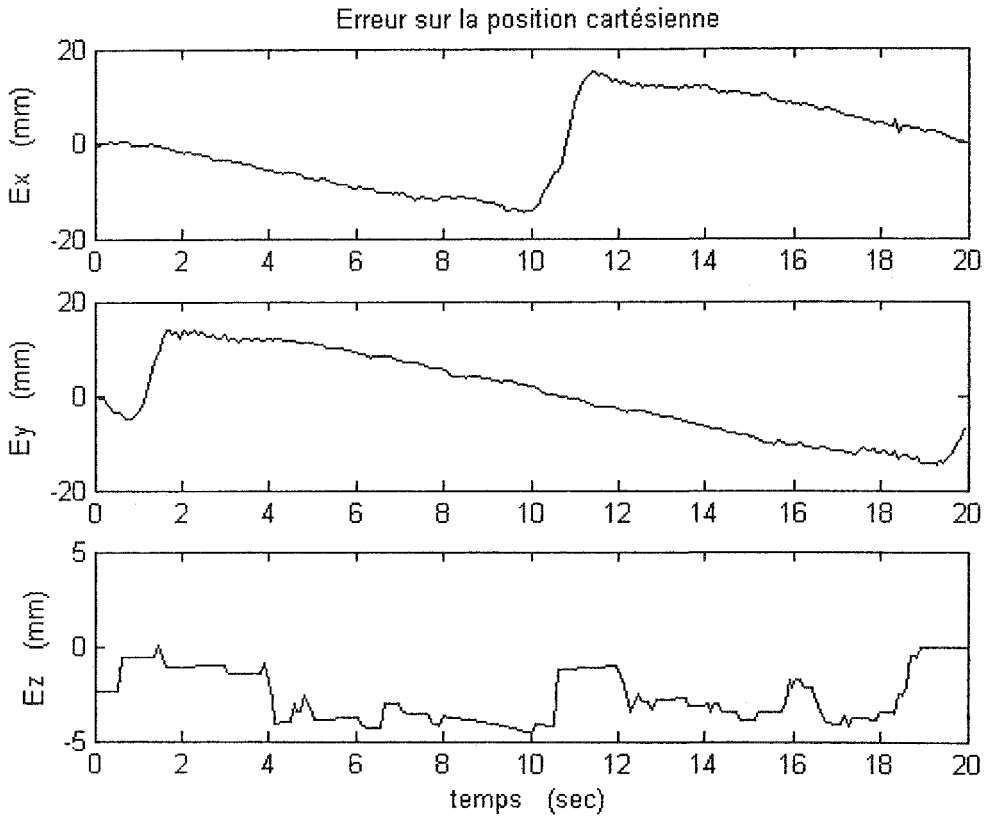


Figure 40 Les erreurs sur les positions cartésiennes du robot en mouvement

4.6.2.5 Asservissement du courant en temps réel

4.6.2.5.1 Principe de contrôle

L'implantation en temps réel d'une loi de commande en robotique, se réalise par un asservissement du couple, et ceci dans un intervalle de temps plus rapide que celui de la position. Nous rappelons que l'asservissement du couple est identique à celui du courrant puisque le rapport entre les deux variables est constant. De plus, l'asservissement du courant se fait à l'aide d'une boucle interne avec un temps

d'échantillonnage (1 ms) plus rapide que celle de la position (10 ms). Et le principe se base sur la capture analogique du courant par la carte S626 qui doit être filtrée par un filtre du 2^{eme} ordre et aussi l'introduction d'un contrôleur PI qui élimine l'erreur par rapport à la consigne. La figure 41 présente les blocs du contrôle du courant d'un axe du robot. De plus, les valeurs utilisées pour le filtrage du courrant sont respectivement $\tau_1=50\ ms$ et $\tau_2=0.159\ ms$.

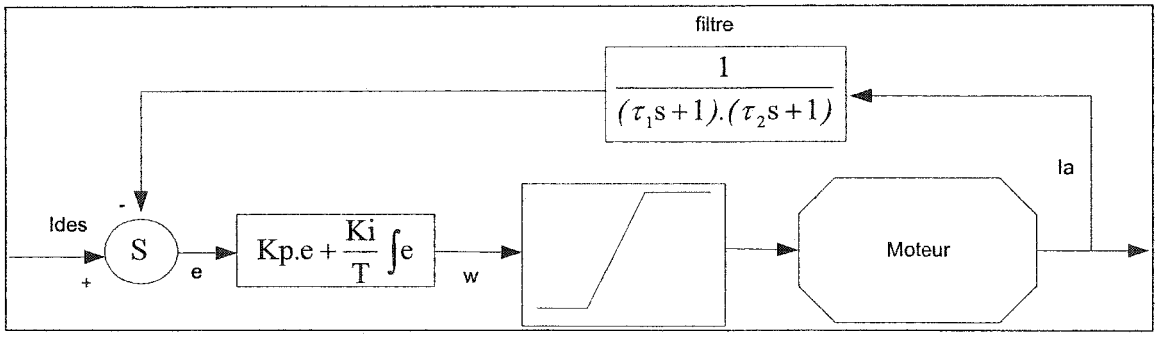


Figure 41 Principe de contrôle du courant

Ensuite, nous avons ajusté les paramètres du gain Kp et Ki, de telle façon à avoir un meilleur comportement. De plus, après plusieurs essais sur le robot, nous avons retenu les valeurs suivantes comme les meilleurs gains d asservissement du courant :

Tableau XIII

Paramètres du gain pour l'asservissement du courant

Axe	1	2	3	4	5
Kp	150	150	65	60	250
Ki	3	4	4	5	20

En se basant sur ces valeurs, nous avons obtenu les résultats décrits dans la partie suivante.

4.6.2.5.2 Résultats expérimentaux

L'expérience faite sur le robot MAKER 100 en temps réel est de prendre comme consigne un signal sinusoïdal pour chaque actionneur du robot. La figure 42 montre, l'évolution du courant filtré et désiré dans les axes du robot pendant un intervalle de temps de 5 secondes.

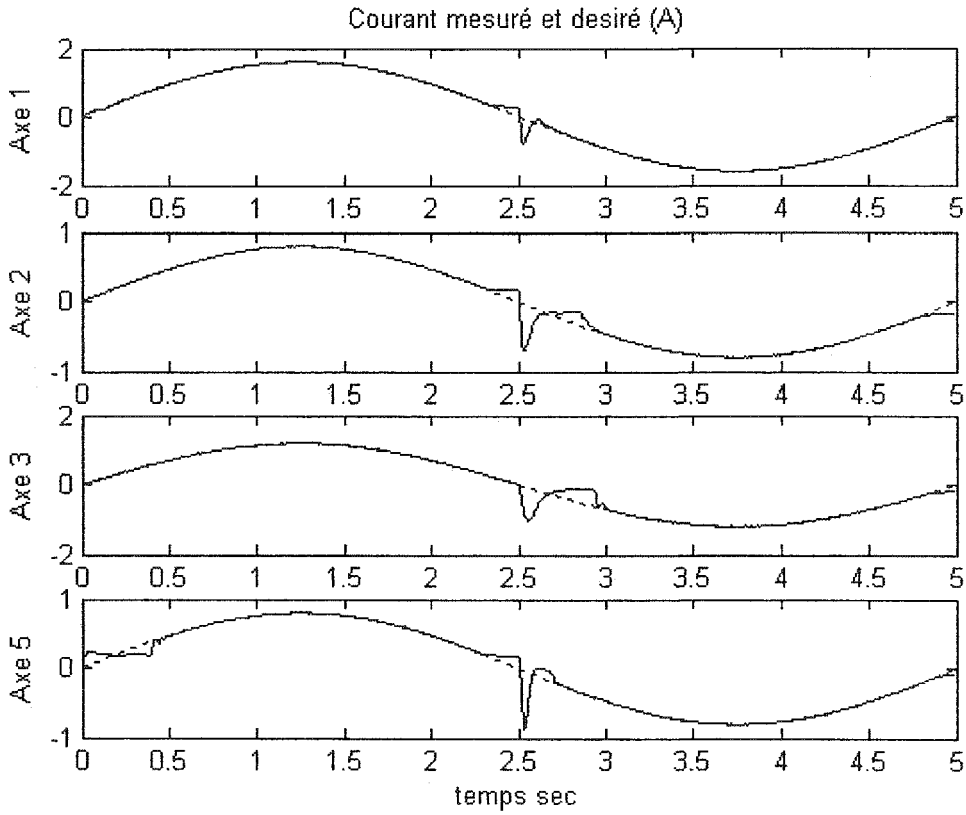


Figure 42 Evolution du courant filtré et désiré des moteurs du robot

Ensuite, la figure 43 montre les erreurs de la poursuite de la consigne en courant pour les actionneurs du robot.

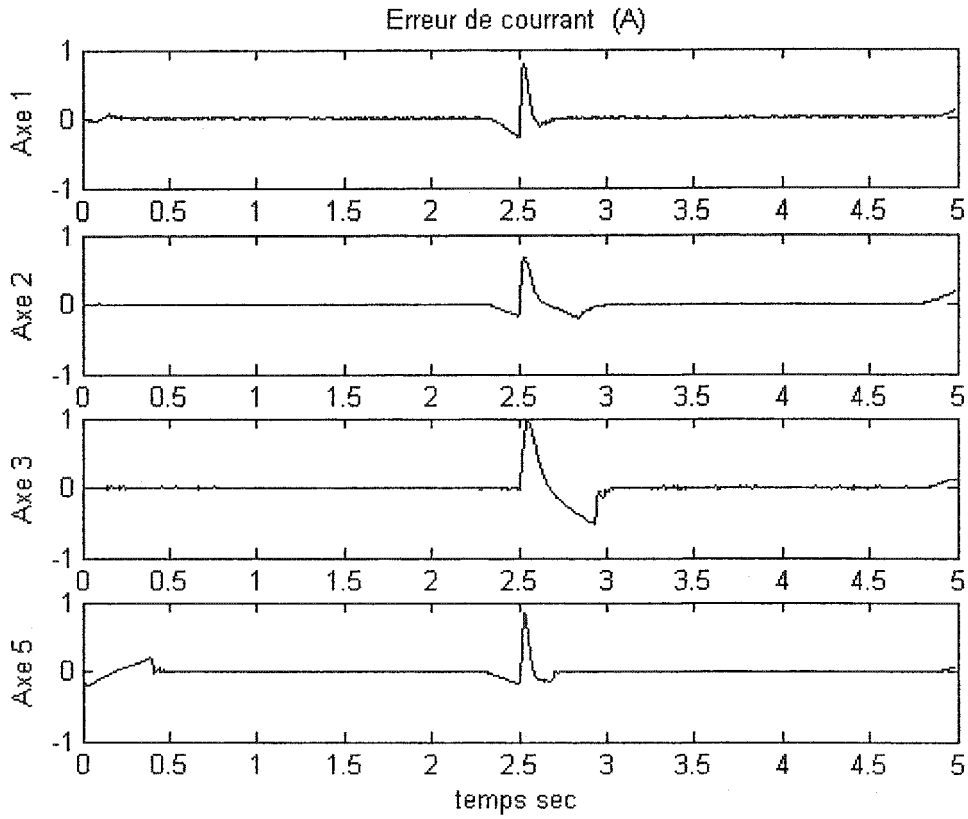


Figure 43 Évolution des erreurs du courant

4.6.2.6 Analyse des résultats

En faisant une étude analytique des résultats de l'implantation en temps réel, nous constatons que malgré la simplicité de la commande conventionnelle PD, elle assure une faible erreur de la poursuite dans les deux espaces articulaires et cartésiens. Aussi, l'asservissement du courant garantit une meilleure poursuite de la consigne particulièrement dans le cas où le signe de la consigne est constant. Cette phase

représente une étape préliminaire importante servant à implanter toute autre commande robotique évoluée en l'occurrence de la commande adaptative directe.

DISCUSSION ET INTERPRETATION DES RESULTATS

La poursuite de trajectoires effectuée par le robot US MAKER 100, a été réalisée avec une performance très satisfaisante à l'aide d'un simple contrôleur en l'occurrence d'un PD. De plus, un asservissent du courrant a été réalisé avec un grand succès. Et il permettra l'implantation d'autres lois de commande robotique.

D'autre part, la planification d'une tâche repose, principalement, sur la génération de la trajectoire désirée qui doit être faite avec les méthodes numériques les plus avancées en calcul matriciel et de lissage pour minimiser l'espace mémoire et le temps de calcul lorsque le robot est en ligne.

De plus, la performance obtenue en temps réel est basée sur des paramètres de gains ajustés manuellement pour assurer un fonctionnement satisfaisant du robot en minimisant l'écart entre les trajectoires désirées et mesurées. Ces paramètres ne représentent pas des valeurs optimales pour le contrôle. En combinant ces deux derniers points, la coopération entre robots se limitera aux contraintes matérielles en fonction du nombre d'axes à contrôler et les cartes E/S implantées.

RECOMMANDATIONS

Pour un meilleur rendement, nous suggérons la disposition et l'exploitation d'autres équipements et accessoires liées à la robotique qui facilitent la réalisation de la tâche dans les meilleures conditions. Ce matériel permet de régler les problèmes de bruits des signaux électriques circulant dans les deux sens de la liaison PC – Robot, causés par plusieurs facteurs (Interactions entres les cartes, Champs magnétiques des moteurs, etc.). Citons, à titre d'exemple, les triggers utilisés pour le filtrage de bruits.

Avec les problèmes de fiabilité inhérents aux premières versions de toutes créations, il est maintenant facile de comprendre que le développement du robot fut si lent et difficile. Il serait à recommander que chaque phase soit premièrement montée sur des versions stables. Ainsi, le développement pourrait se faire plus rapidement, en ne portant que des modules stables aux réalisations ultérieures.

De plus, l'utilisation de capteurs de force/couple peut être très utile en robotique pour des assemblages très précis, où une très petite erreur de positionnement peut causer des dommages fonctionnels importants aux éléments assemblés.

De plus, la coopération peut être très utile pour remplacer les opérateurs dans des tâches répétitives et pénibles. En règle générale, l'utilisation d'un système de coopération devient nécessaire dès que l'opération à effectuer nécessite un retour d'informations sur les états de charges en bout d'outil, à fin d'éviter d'endommager le robot et les pièces à manipuler.

D'autre part, l'algorithme de Taylor garantit que les points intermédiaires restent dans la ligne rectiligne de l'espace cartésien, mais, si l'enveloppe de travail du robot représente

un espace non convexe on doit rajouter la contrainte d'appartenance à cet espace lors de la génération de la trajectoire.

Et ensuite, entre deux points générés, il peut avoir que le polynôme de lissage dépasse les limites du joint, c'est pour cela, qu'il faut s'assurer que la trajectoire désirée soit à l'intérieur des limites physiques de chaque joint.

D'un autre coté, les mises à zéro mécaniques du robot se faisaient, jusqu'à présent, de façon manuelle à chaque début d'expérience, alors que nous proposons une installation des fins de courses au bout des limites des joints pour signaler au programme d'initialisation, la mise à zéro automatique.

CONCLUSION

Ayant travaillé pour ce projet dans un laboratoire au sein de l'Ecole de Technologie Supérieure, nous avons divisé le travail en différentes sections, donnant évidemment priorité à la construction d'un prototype, à l'aide de Simulink de Matlab, sur lequel pourraient être testés les divers modules que nous ajouterions au robot. Nous avons d'abord élaboré un contrôleur PD simple, puis une commande adaptative. De plus, une interface graphique sous QNX fut montée. Le design modulaire du robot facilite l'intégration des différentes composantes.

Il est à noter que certaines composantes du robot représentaient pour nous une nouvelle technologie qui demanderait forcément plus de temps de développement. Plusieurs sections informatiques furent allégées, rendant le développement et les tests beaucoup plus simples. Il est enrichissant de savoir que, malgré toutes les contraintes, qu'elles soient budgétaires, temporelles, matérielles ou autres, nous sommes parvenu à contrôler en temps réel un robot qui se rapprochait à réaliser des tâches prédéfinies initialement. Évidemment, comme dans tout projet, il y eu des imprévus, et nous pouvons tout de même résumer le projet comme suit. Un robot fut contrôlé, ayant pour but de réaliser le parcours d'un triangle avec une meilleure performance avec toutes les évolutions nécessaires. Il est apparent que le robot est capable d'évoluer de façon fiable, et à ce niveau là, nous pouvons dire que le contrôle de deux robots en coopération est aisément réalisable. C'est juste il faut remplacer le modèle dynamique du robot par celui des robots et l'objet. Cet ensemble pourrait être considéré comme un seul robot mais avec des contraintes géométriques.

ANNEXES

Modélisation DU ROBOT US MAKER 100

A.1 Matrices de transformation homogène

En général la matrice de transformation homogène s'écrit de la façon suivante :

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ c\alpha_{i-1}s\theta_i & c\alpha_{i-1}c\theta_i & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\alpha_{i-1}s\theta_i & s\alpha_{i-1}c\theta_i & c\alpha_{i-1} & -d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(A.1)} $$

Dans la partie qui va suivre on va déterminer la matrice de transformation homogène qui correspond à chaque joint : ${}^{0}_{1}T$; ${}^{1}_{2}T$; ${}^{2}_{3}T$; ${}^{3}_{4}T$; ${}^{4}_{5}T$.

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(A.2)} $$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(A.3)} $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & d_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4} $$

$$ {}^{3}_{4}T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_4 & -c\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5} $$

$$ {}^{4}_{5}T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5} $$

A.2 Étude de la cinématique directe

Pour étudier la cinématique directe, il faut tout d'abord déterminer la matrice de transformation globale du robot définie comme suit.

$$ {}^{0}_{5}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T * {}^{4}_{5}T \tag{A.6} $$

On trouve après simplification et en posant :

$cos\ (\theta_i) = c_i = c\theta_i;$

$sin\ (\theta_i) = s_i = s\theta_i;$

$cos(\theta_i+\theta_j) = c_{ij}$ *et* $sin(\theta_i+\theta_j) = s_{ij},$

$$ {}^{0}_{5}T = \begin{bmatrix} c_1.c_5.c_{24} - s_1.s_5 & -c_1.s_5.c_{24} - s_1.c_5 & -c_1.s_{24} & -c_1.s_2.d_3 \\ s_1.c_5.c_{24} + c_1.s_5 & -s_1.s_5.c_{24} + c_1.c_5 & -s_1.s_{24} & -s_1.s_2.d_3 \\ c_5.s_{24} & -s_5.s_{24} & c_{24} & c_2.d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.7} $$

De plus, le point du contact Pc de l'objet a les coordonnées suivantes par rapport au repère {5} : Pc=[0 0 l_5]^T; avec l_5 est la distance du point Pc à l'origine du repère {5}.

On sait que :

$$ {}^{5}_{c}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Ce qui nous donne l'expression résultante suivante :

$$ {}^{0}_{c}T = {}^{0}_{5}T.{}^{5}_{c}T = \begin{bmatrix} c_1.c_5.c_{24} - s_1.s_5 & -c_1.s_5.c_{24} - s_1.c_5 & -c_1.s_{24} & -c_1.(s_2.d_3 + s_{24}.l_5) \\ s_1.c_5.c_{24} + c_1.s_5 & -s_1.s_5.c_{24} + c_1.c_5 & -s_1.s_{24} & -s_1.(s_2.d_3 + s_{24}.l_5) \\ c_5.s_{24} & -s_5.s_{24} & c_{24} & c_2.d_3 + c_{24}.l_5. \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.8)} $$

A.3 Étude de la cinématique inverse

Afin d'étudier la cinématique inverse, on va assumer qu'on a un point dans l'espace défini par une matrice 4 par 4 de la même forme que la matrice qu'on a déterminé en étudiant la cinématique directe. Cette matrice définit les coordonnées (position et orientation) de ce point dans l'espace.

$$ M = \begin{matrix} \\ m_x \\ m_y \\ m_z \\ 0 \end{matrix} \begin{matrix} \begin{matrix} m & n & o & p \end{matrix} \\ \begin{bmatrix} n_x & o_x & p_x \\ n_y & o_y & p_y \\ n_z & o_z & p_z \\ 0 & 0 & 1 \end{bmatrix} \end{matrix} \quad \text{(A.9)} $$

Sachant que M et ${}^{0}_{c}T$ ont la même forme, on veut déterminer les paramètres (θ_1, θ_2, θ_4, θ_5 et d_3) du robot qui permettent d'atteindre ce point.

Autrement, en identifiant la matrice d'orientation ${}^{0}_{c}R$ de M avec celle d'Euler, on tire le système suivant :

$$\varphi = \theta_1 \qquad \psi = \theta_2 + \theta_4 \quad \gamma = \theta_5 \tag{A.10}$$

Ensuite, en égalant les cordonnées cartésiennes exprimées dans le repère de base {0}, on trouve les relations suivantes :

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} -c_1.(s_2.d_3 + s_{24}.l_5) \\ -s_1.(s_2.d_3 + s_{24}.l_5) \\ c_2.d_3 + c_{24}.l_5 \end{bmatrix} \tag{A.11}$$

En combinant les systèmes A.10 et A.11, on aboutit au système A.12 suivant :

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ \psi \\ \gamma \end{bmatrix} = \begin{bmatrix} -c_1.(s_2.d_3 + s_{24}.l_5) \\ -s_1.(s_2.d_3 + s_{24}.l_5) \\ c_2.d_3 + c_{24}.l_5 \\ \theta_2 + \theta_4 \\ \theta_5. \end{bmatrix} \tag{A.12}$$

$$\Rightarrow {p_x}^2 + {p_y}^2 = (s_2.d_3 + s_\psi.l_5)^2.$$

$$\Rightarrow s_2.d_3 + s_\psi.l_5 = \varepsilon.\sqrt{{p_x}^2 + {p_y}^2}$$

$$\Rightarrow s_2.d_3 = \varepsilon.\sqrt{{p_x}^2 + {p_y}^2} - s_\psi.l_5. \text{ avec } \varepsilon = \pm 1. \tag{A.13}$$

De plus, d'après A.12, on a :

$$p_z = c_2.d_3 + c_\psi.l_5 \Rightarrow c_2.d_3 = p_z - c_\psi.l_5. \tag{A.14}$$

En exploitant les équations A.13 et A.14, on déduit :

$$d_3 = \left[(p_z - c_\psi.l_5)^2 + (\varepsilon\sqrt{{p_x}^2 + {p_y}^2} - s_\psi.l_5)^2 \right]^{1/2}. \tag{A.15}$$

En connaissant d_3, les équations A.13 et A.14, permettent de déduire θ_2 :

$$\theta_2 = A\tan 2(\varepsilon\sqrt{p_x{}^2+p_y{}^2} - s_\psi.l_5, p_z - c_\psi.l_5). \tag{A.16}$$

une fois calculées les valeurs d_3 et θ_2, les autres valeurs se déduisent d'après le système A.12, comme suit :

$$\begin{bmatrix}\theta_1\\ \theta_4\\ \theta_5\end{bmatrix} = \begin{bmatrix} A\tan 2(\frac{p_y}{Ac}, \frac{p_x}{Ac}) \\ \psi - \theta_2 \\ \gamma. \end{bmatrix} \tag{A.17}$$

avec :

$\varepsilon = \mp 1.$

$Ac = -(s_2.d_3 + s_\psi.l_5).$

A.4 Vitesse angulaires et linéaires du Robot

Dans le cas d'une articulation rotative les vitesses angulaires W et linéaires V sont données par les formules suivantes :

$${}^{i+1}W_{i+1} = {}^{i+1}_{i}R.{}^{i}W_i + \dot{\theta}_{i+1}.{}^{i+1}Z_{i+1} \tag{A.18}$$

$${}^{i+1}V_{i+1} = {}^{i+1}_{i}R.({}^{i}V_i + {}^{i}W_i \otimes {}^{i}P_{i+1}) \tag{A.19}$$

Dans le cas d'une articulation prismatique les vitesses angulaires et linéaires sont données par les formules suivantes :

$${}^{i+1}W_{i+1} = {}^{i+1}_{i}R.{}^{i}W_i. \tag{A.20}$$

$${}^{i+1}V_{i+1} = {}^{i+1}_{i}R.({}^{i}V_i + {}^{i}W_i \otimes {}^{i}P_{i+1} + \dot{d}_{i+1}.{}^{i+1}Z_{i+1}) \tag{A.21}$$

avec :

${}^{i+1}_{i}R.$ est le transposé de la matrice d'orientation ${}^{i}_{i+1}R.$ déduite de la matrice ${}^{i}_{i+1}T.$

${}^{i}P_{i+1}$ est l'origine du repère $\{i+1\}$ exprimé dans le repère $\{i\}$.

Pour chaque articulation, on applique ces formules pour aboutir aux résultats suivants :

A.4.1 Articulation 1 : rotatif

On a :

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.22} $$

$$ {}^{1}W_1 = {}^{1}_{0}R.{}^{0}W_0 + \dot{\theta}_1.{}^{1}Z_1 = \dot{\theta}_1.{}^{1}Z_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \tag{A.23} $$

$$ {}^{1}V_1 = {}^{1}_{0}R.({}^{0}V_0 + {}^{0}W_0 \otimes {}^{0}P_1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.24} $$

A.4.2 Articulation 2 : rotatif

On a :

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.25} $$

$$ {}^{1}_{2}R = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 \\ 0 & 0 & -1 \\ s\theta_2 & c\theta_2 & 0 \end{bmatrix} \Rightarrow {}^{2}_{1}R = \begin{bmatrix} c\theta_2 & 0 & s\theta_2 \\ -s\theta_2 & 0 & c\theta_2 \\ 0 & -1 & 0 \end{bmatrix} \tag{A.26} $$

Et ensuite les vitesses se déterminent comme suit :

$$ {}^{2}W_2 = {}^{2}_{1}R.{}^{1}W_1 + \dot{\theta}_2.{}^{2}Z_2 = \begin{bmatrix} c_2 & 0 & s_2 \\ -s_2 & 0 & c_2 \\ 0 & -1 & 0 \end{bmatrix}.\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1.s_2 \\ \dot{\theta}_1.c_2 \\ \dot{\theta}_2 \end{bmatrix} \tag{A.27} $$

$$ {}^{2}V_{2} = {}^{2}_{1}R.({}^{1}V_{1} + {}^{1}W_{1} \otimes {}^{1}P_{2}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.28} $$

A.4.3 Articulation 3 : prismatique

On a :

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & d_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.29} $$

$$ {}^{2}_{3}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \Rightarrow {}^{3}_{2}R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \tag{A.30} $$

Les vitesses sont donc :

$$ {}^{3}W_{3} = {}^{3}_{2}R.{}^{2}W_{2} = \begin{bmatrix} \dot{\theta}_1.s_2 \\ \dot{\theta}_2 \\ -\dot{\theta}_1.c_2 \end{bmatrix} \tag{A.31} $$

$$ {}^{3}V_{3} = {}^{3}_{2}R.({}^{2}V_{2} + {}^{2}W_{2} \otimes {}^{2}P_{3} + \dot{d}_3.{}^{3}Z_{3}) = \begin{bmatrix} -d_3.\dot{\theta}_2 \\ d_3.\dot{\theta}_1.s_2 \\ \dot{d}_3 \end{bmatrix} \tag{A.32} $$

A.4.4 Articulation 4 : rotative

On a :

$$ {}^{3}_{4}T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_4 & -c\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.33} $$

$$ {}^{3}_{4}R = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 \\ 0 & 0 & 1 \\ -s\theta_4 & -c\theta_4 & 0 \end{bmatrix} \Rightarrow {}^{4}_{3}R = \begin{bmatrix} c\theta_4 & 0 & -s\theta_4 \\ -s\theta_4 & 0 & -c\theta_4 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{(A.34)} $$

Les vitesses sont données par :

$$ {}^{4}W_4 = {}^{4}_{3}R.{}^{3}W_3 + \dot{\theta}_4.{}^{4}Z_4 = \begin{bmatrix} \dot{\theta}_1.s_{24} \\ \dot{\theta}_1.c_{24} \\ \dot{\theta}_2 + \dot{\theta}_4 \end{bmatrix} \quad \text{(A.35)} $$

$$ {}^{4}V_4 = {}^{4}_{3}R.({}^{3}V_3 + {}^{3}W_3 \otimes {}^{3}P_4) = \begin{bmatrix} -d_3.\dot{\theta}_2.c_4 - \dot{d}_3.s_4 \\ -d_3.\dot{\theta}_2.s_4 - \dot{d}_3.c_4 \\ d_3.\dot{\theta}_1.s_2 \end{bmatrix} \quad \text{(A.36)} $$

A.4.5 Articulation 5 : rotative

On a : $$ {}^{4}_{5}T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}R = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 \\ 0 & 0 & 1 \\ -s\theta_5 & -c\theta_5 & 0 \end{bmatrix} \Rightarrow {}^{5}_{4}R = \begin{bmatrix} c\theta_5 & 0 & -s\theta_5 \\ -s\theta_5 & 0 & -c\theta_5 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{(A.37)} $$

Et ensuite les vitesses se déterminent comme suit :

$$ {}^{5}W_5 = {}^{5}_{4}R.{}^{4}W_4 + \dot{\theta}_5.{}^{5}Z_5 = \begin{bmatrix} c_5.\dot{\theta}_1.s_{24} - s_5.(\dot{\theta}_2 + \dot{\theta}_4) \\ -s_5.\dot{\theta}_1.s_{24} - c_5.(\dot{\theta}_2 + \dot{\theta}_4) \\ \dot{\theta}_5 + \dot{\theta}_1.c_{24} \end{bmatrix} \quad \text{(A.38)} $$

$$ {}^{5}V_5 = {}^{5}_{4}R.({}^{4}V_4 + {}^{4}W_4 \otimes {}^{4}P_5) = \begin{bmatrix} (-d_3.\dot{\theta}_2.c_4 - \dot{d}_3.s_4).c_5 - d_3.\dot{\theta}_1.s_2.s_5 \\ (d_3.\dot{\theta}_2.c_4 + \dot{d}_3.s_4).s_5 - d_3.\dot{\theta}_1.s_2.c_5 \\ -d_3.\dot{\theta}_2.s_4 - \dot{d}_3.c_4 \end{bmatrix} \quad \text{(A.39)} $$

A.4.6 Vitesse angulaire du poignet

Elle donnée par :

$$
\begin{aligned}
{}^{0}W_5 &= {}^{0}_{5}R.{}^{5}W_5 \\
&= \begin{bmatrix} c_1.c_5.c_{24} - s_1.s_5 & -c_1.s_5.c_{24} - s_1.c_5 & -c_1.s_{24} \\ s_1.c_5.c_{24} + c_1.s_5 & -s_1.s_5.c_{24} + c_1.c_5 & -s_1.s_{24} \\ c_5.s_{24} & -s_5.s_{24} & c_{24} \end{bmatrix} \begin{bmatrix} c_5.\dot{\theta}_1.s_{24} - s_5.(\dot{\theta}_2 + \dot{\theta}_4) \\ -s_5.\dot{\theta}_1.s_{24} - c_5.(\dot{\theta}_2 + \dot{\theta}_4) \\ \dot{\theta}_5 + \dot{\theta}_1.c_{24} \end{bmatrix} \\
&= \begin{bmatrix} s_1.\dot{\theta}_2 + s_1.\dot{\theta}_4 - c_1.s_{24}.\dot{\theta}_5. \\ -c_1.\dot{\theta}_2 - c_1.\dot{\theta}_4 - s_1.s_{24}.\dot{\theta}_5. \\ .\dot{\theta}_1 + c_{24}.\dot{\theta}_5. \end{bmatrix} = \begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix}
\end{aligned}
\tag{A.40}
$$

A.4.7 Vitesse linéaire du poignet

La vitesse linéaire du poignet est déterminée soit en appliquant la même formule mais cette fois avec des vitesses linéaires.

$$
\begin{aligned}
{}^{0}V_5 &= {}^{0}_{5}R.{}^{5}V_5 \\
&= \begin{bmatrix} c_1.c_5.c_{24} - s_1.s_5 & -c_1.s_5.c_{24} - s_1.c_5 & -c_1.s_{24} \\ s_1.c_5.c_{24} + c_1.s_5 & -s_1.s_5.c_{24} + c_1.c_5 & -s_1.s_{24} \\ c_5.s_{24} & -s_5.s_{24} & c_{24} \end{bmatrix} \begin{bmatrix} (-d_3.\dot{\theta}_2.c_4 - \dot{d}_3.s_4).c_5 - d_3.\dot{\theta}_1.s_2.s_5 \\ (d_3.\dot{\theta}_2.c_4 + \dot{d}_3.s_4).s_5 - d_3.\dot{\theta}_1.s_2.c_5 \\ -d_3.\dot{\theta}_2.s_4 - \dot{d}_3.c_4 \end{bmatrix} \\
&= \begin{bmatrix} s_1.s_2.d_3.\dot{\theta}_1 - c_1.c_2.d_3.\dot{\theta}_2 - c_1.s_2.\dot{d}_3 \\ -c_1.s_2.d_3.\dot{\theta}_1 - s_1.c_2.d_3.\dot{\theta}_2 - s_1.s_2.\dot{d}_3 \\ -s_2.d_3.\dot{\theta}_2 + c_2.\dot{d}_3 \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}
\end{aligned}
\tag{A.41}
$$

A 5 Génération de la trajectoire

A.5.1 Technique d'interpolation

La technique d'interpolation consiste à interpoler entre les points intermédiaires pour produire une trajectoire souple (ou lisse), c'est à dire qu'elle admet au moins deux dérivées continues pour éviter le cas d'une accélération infinie. Pour satisfaire à cette exigence, on peut adopter les trajectoires cubiques qui ont deux dérivées.

A.5.2 Trajectoires cubiques avec deux points intermédiaires

Prenons le cas le plus simple, le point de départ q^0 et le point d'arrivée q^f et deux points intermédiaires q^1 et q^2. Une trajectoire cubique (dans l'espace des articulations) est donnée par:

$$q(t) = a_0 + a_1.t + a_2.t^2 + a_3.t^3 \quad \text{avec} \quad 0 \leq t \leq T$$

Où T est le temps nécessaire pour parcourir cette trajectoire.

Puisqu'on a quatre points on a besoin de trois trajectoires cubiques l'une $q_1(t)$ pour aller du point de départ q_0 au point q_1; la seconde, $q_2(t)$ pour aller de q_1 au point q_2; et la dernière trajectoire $q_3(t)$ pour aller de q_2 au point q_f:
Première trajectoire

$$\begin{cases} q_1(t) = a_{10} + a_{11}.t + a_{12}.t^2 + a_{13}.t^3. \\ \dot{q}_1(t) = a_{11} + 2.a_{12}.t + 3.a_{13}.t^2. \\ \ddot{q}_1(t) = 2.a_{12} + 6.a_{13}.t. \end{cases} \tag{A.42}$$

Avec $0 \leq t \leq t_1$ où t_1 est le temps nécessaire pour aller de q_0 à q_1

Seconde trajectoire :

$$\begin{cases} q_2(t) = a_{20} + a_{21}.t + a_{22}.t^2 + a_{23}.t^3. \\ \dot{q}_2(t) = a_{21} + 2.a_{22}.t + 3.a_{23}.t^2. \\ \ddot{q}_2(t) = 2.a_{22} + 6.a_{23}.t. \end{cases} \quad \text{(A.43)}$$

avec $0 \leq t \leq t_2$ où t_2 est le temps nécessaire pour aller de q_1 à q_2

Troisième trajectoire :

$$\begin{cases} q_3(t) = a_{30} + a_{31}.t + a_{32}.t^2 + a_{33}.t^3. \\ \dot{q}_3(t) = a_{31} + 2.a_{32}.t + 3.a_{33}.t^2. \\ \ddot{q}_3(t) = 2.a_{32} + 6.a_{33}.t. \end{cases} \quad \text{(A.44)}$$

avec $0 \leq t \leq t_3$ où t_3 est le temps nécessaire pour aller de q_2 à q_f

Le temps total pour aller du point de départ au point final est alors :

$$t_{traj} = t_1 + t_2 + t_3 \quad \text{(A.45)}$$

Puisqu'on connaît la vitesse initiale v_0 qui est la vitesse au point q_0 et la vitesse finale v_f qui est la vitesse au point q_f et en assurant la continuité de vitesse et celle de l'accélération aux points intermédiaires, on peut calculer les coefficients des équations des trajectoires. En effet :

$$q_1(0) = a_{10} \Rightarrow a_{10.} = q^0$$
$$q_1(t_1) = q_2(0) \Rightarrow a_{10.} + a_{11.}t_1 + a_{12.}t_1^2 + a_{13.}t_1^3 = q^1$$
$$\Rightarrow a_{11.}t_1 + a_{12.}t_1^2 + a_{13.}t_1^3 = q^1 - q^0.$$
$$\dot{q}_1(t_1) = \dot{q}_2(0) \Rightarrow a_{11.} + 2.a_{12.}t_1 + 3.a_{13.}t_1^2 = 0$$
$$\ddot{q}_1(t_1) = \ddot{q}_2(0) \Rightarrow a_{11.} + 3.a_{12.}t_1 - .a_{22.} = 0$$

$$q_2(0) = a_{20} \Rightarrow a_{20.} = q^1.$$
$$q_2(t_2) = q_3(0) \Rightarrow a_{20.} + a_{21..}t_2 + a_{22}.t_2^2 + a_{23..}t_2^3 = q^2$$
$$\Rightarrow a_{21..}t_2 + a_{22..}t_2^2 + a_{23..}t_2^3 = q^2 - q^1.$$
$$\dot{q}_2(t_2) = \dot{q}_3(0) \Rightarrow a_{21.} + 2.a_{22..}t_2 + 3.a_{23..}t_2^3 = 0$$
$$\ddot{q}_2(t_2) = \ddot{q}_3(0) \Rightarrow a_{21.} + 3.a_{22..}t_2 - .a_{32.} = 0$$

$$q_3(0) = a_{30} \Rightarrow a_{30.} = q^2$$
$$q_3(t_3) = q^f \Rightarrow a_{30.} + a_{31..}t_3 + a_{32..}t_3^2 + a_{23..}t_3^3 = q^f$$
$$\Rightarrow a_{31..}t_3 + a_{32..}t_3^2 + a_{33..}t_3^3 = q^f - q^2 \qquad \text{(A.46)}$$

$$\dot{q}_3(t_3) = v^f \Rightarrow a_{31.} + 2.a_{32..}t_3 + 3.a_{33..}t_3^2 = v^f$$
$$\dot{q}_1(0) = v^0 \Rightarrow a_{11.} = v^0$$

soit sous forme matricielle on a:

$$\begin{bmatrix} v_0 \\ q_1 - q_0 \\ 0 \\ 0 \\ q_2 - q_1 \\ 0 \\ 0 \\ 0 \\ q_f - q_2 \\ v_f \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_1 & t_1^2 & t_1^3 & & & & & & 0 \\ 1 & 2.t_1 & 3.t_1^2 & -1 & & & & & 0 \\ 0 & 1 & 3.t_1 & 0 & -1 & & & & 0 \\ 0 & & 0 & t_2 & t_2^2 & t_2^3 & & & 0 \\ 0 & & & 1 & 2.t_2 & 3.t_2^2 & -1 & & 0 \\ 0 & & & & 1 & 3.t_2 & 0 & -1 & 0 \\ 0 & & & & & 0 & t_3 & t_3^2 & t_3^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2.t_3 & 3.t_3^2 \end{pmatrix} * \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} \qquad \text{(A.47)}$$

A.5.3 Trajectoires cubiques avec plusieurs points intermédiaires

On peut généraliser les formules utilisées pour le problème de deux points intermédiaires. Supposons qu'on a f trajectoires (c'est à dire on a f-1 points intermédiaires). L'équation de la i-ème trajectoire $q_i(t)$ qui doit aller du point q_{i-1} au point q_i est donnée par :

$$q_i(t) = a_{i0} + a_{i1}.t + a_{i2}.t^2 + a_{i3}.t^3 \qquad i=1...f; \tag{A.48}$$

avec $0 \leq t \leq t_i$ où t_i est le temps nécessaire pour parcourir cette trajectoire.
Et i allant de 1 jusqu'à f.

Alors, le temps total pour aller du point de départ au point final est :

$$t_{traj} = \sum_{i=1}^{i=f} ti \tag{A.49}$$

On peut généraliser les équations des trajectoires ainsi que les dérivées primaires et secondaires de la façon suivante :

Pour i entre 1 et (f-1)

$$\begin{cases} q_i(0) = a_{10} \Rightarrow a_{10.} = q^0 \\ q_i(t_i) = q_{i+1}(0) \Rightarrow a_{i1}.t_i + a_{i2}.t_i^2 + a_{i3}.t_i^3 = q^i - q^{i-1} \\ \dot{q}_i(t_i) = \dot{q}_{i+1}(0) \Rightarrow a_{i1.} + 2.a_{i2}.t_i + 3.a_{i3}.t_i^2 = 0 \\ \ddot{q}_i(t_i) = \ddot{q}_{i+1}(0) \Rightarrow a_{i1.} + 3.a_{i2}.t_i - .a_{(i+1).2.} = 0 \end{cases} \tag{A.50}$$

Pour la dernière trajectoire

$$q_f(t_f) = q^f \quad \Rightarrow a_{f1}.t_f + a_{f2}.t_f^2 + a_{f3}.t_f^3 = q^f - q^{f-1} \tag{A.51}$$

En plus, les vitesses initiale et finale sont données par :

$$\dot{q}_f(t_f) = v^f \Rightarrow a_{f1.} + 2.a_{f2}.t_f + 3.a_{f3}.t_f^2 = v^f \tag{A.52}$$
$$\dot{q}_1(0) = v^0 \Rightarrow a_{11.} = v^0$$

$$\begin{bmatrix} v_0 \\ q^1 - q^0 \\ 0 \\ 0 \\ \\ q^i - q^{i-1} \\ 0 \\ \dots \\ \dots \\ \dots \\ \dots \\ \dots \\ q^f - q^{f-1} \\ v^f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & & & & & & & & \\ t_1 & t_1^2 & t_1^3 & 0 & 0 & \dots & & \dots & & \dots & & \dots & \dots \\ 1 & 2t_1 & 3.t_1^2 & -1 & 0 & & & & & & & & \\ 0 & 1 & 3t_1 & 0 & -1 & \dots & & & & & & \dots & \\ & & & & & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & \dots & & & & \dots & \dots & ti & t_1^2 & t_1^2 & 0 & 0 & \dots & & \dots. \\ & & & & & \dots & \dots & 1 & 2ti & 3.t_1^2 & -1 & 0 & \dots \\ & \dots. & & & & \dots & \dots & 0 & 1 & 3ti & 0 & -1 & \dots \\ & & & & & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & \dots. & & & & \dots & & & & & \dots & & \\ & & & & & \dots & & & & & & & \\ & \dots & & & & \dots & & & & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & & \dots & & & & \dots & \dots & 0 & 1 & 3tf & 0 & -1 \\ & \dots & & & & \dots & & & & \dots & \dots & 0 & 0 & tf & t_f^2 & t_f^2 \\ & & & & & \dots & & & & \dots. & \dots & 0 & 0 & 1 & 2tf & 3.t_f^2 \end{bmatrix} * \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ \\ a_{i1} \\ a_{i2} \\ a_{i3} \\ \\ \\ \\ \\ a_{n1} \\ a_{n2} \\ a_{n3} \end{bmatrix}$$

(A.53)

Sous forme matricielle, on peut écrire que $B = A.X$; Où

X est le vecteur des coefficients des trajectoires de dimension $3f$ x 1.

A est une matrice de dimension $3f$ x $3f$

B est un vecteur de données initiales de dimension 3fx1

Donc $X = A^{-1} .B$.

Programmation sous QNX

Brève introduction

Pour développer un projet sous QNX, il faut suivre les étapes suivantes :

Accéder au système via un mot de passe et un nom d'usager.

Éditer un programme en langage C standard à l'aide de l'éditeur

(>> applications >> editor).

Pour compiler ce programme, il faut tout d'abord avoir créer un terminal

(>> applications >> terminal).

Ensuite, taper $ gcc nom_du_fichier –o fichier_executable

Pour l'exécuter, il suffit de taper : $ ./fichier_executable

Pour monter une disquette taper :

mount – t dos /dev/fd0 disquette (un répertoire nommé disquette va être crée)

cd disquette

Pour faire une interface, il vaut mieux utiliser le builder :

(Launch >> Developpement >> Builder) le reste est comme le Visual Basic

Programme de simulation de la commande adaptative directe

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//      PROGRAMME DE LA SIMULATION DE LA COMMANDE ADAPTATIVE DIRECTE
//      APPLIQUE AU ROBOT US MAKER 100 A 5 AXES
//      CE PROGRAMME ENVOIT LES DONNEES DU RESULTAT A L INTERFACE GRAPHIQUE
//      FAIT PAR ABDELKADER HASSANE POUR LA MAITRISE EN GPA A L ETS (2001-2002)
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

Après initialisation et lecture des paramètres de contrôle, on fait appel à la méthode numérique de Runge-Kutta d'ordre 4 :

```
//////// APPEL A LA METHODE DE RUNGE KUTA ////////////////////////////////////
// variables : yf : vecteur d'erreur de position, de vitesse, et paramètres estimés
// cette méthode intègre le système différentiel dyf/dt =zf=f(yf, t)
{
 for(i=0; i<NR; i++) { k1[i]=h*zf[i];  y1[i]=yf[i]+k1[i]/2; }  t1=tps+h/2;  mak5( t1, y1, z2, ep, couple,cpa);
 for(i=0; i<NR; i++) { k2[i]=h*z2[i];  y2[i]=yf[i]+k2[i]/2; }  t2=tps+h/2;  mak5( t2, y2, z3, ep, couple,cpa);
 for(i=0; i<NR; i++) { k3[i]=h*z3[i];  y3[i]=yf[i]+k3[i]; }  t3=tps+h;  mak5( t3, y3, z4, ep, couple,cpa);
 for(i=0; i<NR; i++) { k4[i]=h*z4[i]; }
 for(i=0; i<NR; i++)  yf[i]+=(k1[i]+2*k2[i]+2*k3[i]+k4[i])/6;;
 tps += h;
}
//////**********************************************************************///////
```

La routine du modèle dynamique (mak5)

```
,// cette procédure calcul la loi de commande a appliquer sous forme du couple. Aussi, elle donne le vecteur de
 // position, de vitesse, et des paramètres estimés a l'instant tps.
// q1pd position désirée de l'axe 1; q1vd vitesse désirée de l'axe 1; q1ad accélération désirée de l'axe 1
// q1pm position mesurée de l'axe 1; q1vm vitesse mesurée de l'axe 1; q1am accélération mesurée de l'axe 1
// qdd vecteur points nécessaires de la trajectoire articulaire
// p1e : paramètre estimé, migm : matrice d'adaptation, cpa : vecteur des paramètres de contrôle
 tps=tu-tem;
 q1pd=qdd[kx][0];  q2pd=qdd[kx][1];  q3pd=qdd[kx][2];  q4pd=qdd[kx][3];  q5pd=qdd[kx][4];
 for (i=0;i<5;i++) {  for (j=0;j<3;j++) {  b[i][j]=bdd[kx][3*i+j];  } }

 pos_i[0]=q1pd;  pos_i[1]=q2pd;  pos_i[2]=q3pd;  pos_i[3]=q4pd;  pos_i[4]=q5pd;

                q1pd=pos_i[0]+b[0][0]*tps+b[0][1]*tps*tps+b[0][2]*tps*tps*tps;
                q2pd=pos_i[1]+b[1][0]*tps+b[1][1]*tps*tps+b[1][2]*tps*tps*tps;
                q3pd=pos_i[2]+b[2][0]*tps+b[2][1]*tps*tps+b[2][2]*tps*tps*tps;
                q4pd=pos_i[3]+b[3][0]*tps+b[3][1]*tps*tps+b[3][2]*tps*tps*tps;
                q5pd=pos_i[4]+b[4][0]*tps+b[4][1]*tps*tps+b[4][2]*tps*tps*tps;

                q1vd=b[0][0]+2.0*b[0][1]*tps+3.0*b[0][2]*tps*tps;
                q2vd=b[1][0]+2.0*b[1][1]*tps+3.0*b[1][2]*tps*tps;
                q3vd=b[2][0]+2.0*b[2][1]*tps+3.0*b[2][2]*tps*tps;
                q4vd=b[3][0]+2.0*b[3][1]*tps+3.0*b[3][2]*tps*tps;
                q5vd=b[4][0]+2.0*b[4][1]*tps+3.0*b[4][2]*tps*tps;
```

```
                    q1ad=2.0*b[0][1]+6.0*b[0][2]*tps;
                    q2ad=2.0*b[1][1]+6.0*b[1][2]*tps;
                    q3ad=2.0*b[2][1]+6.0*b[2][2]*tps;
                    q4ad=2.0*b[3][1]+6.0*b[3][2]*tps;
                    q5ad=2.0*b[4][1]+6.0*b[4][2]*tps;
for(i=0;i<19; i++)  yes[i]=yf[i];
 p1e = yes[0];  p2e = yes[1];  p3e = yes[2];  p4e = yes[3];  p5e = yes[4];  p6e = yes[5];  p7e = yes[6];
 p8e = yes[7];  p9e = yes[8];  p10e = yes[9];  p11e = yes[10];  p12e = yes[11];  p13e = yes[12];
p14e = yes[13];  p15e = yes[14];  p16e = yes[15];  p17e = yes[16];  p18e = yes[17];  p19e = yes[18];

 q1pm=yf[19]+q1pd; q2pm=yf[20]+q2pd; q3pm=yf[21]+q3pd; q4pm=yf[22]+q4pd; q5pm=yf[23]+q5pd;
 q1vm=yf[24]+q1vd; q2vm=yf[25]+q2vd; q3vm=yf[26]+q3vd; q4vm=yf[27]+q4vd; q5vm=yf[28]+q5vd;

 m11=1;m22=1;m33=1;m44=1;m55=1;
 for (i=0; i<NB_PAR;i++){  kii =i+10;    migm[i] =  -cpa[kii]; }

 q1vr=q1vd-ld1*(q1pm-q1pd);  q1ar=q1ad-ld1*(q1vm-q1vd);
 q2vr=q2vd-ld2*(q2pm-q2pd);  q2ar=q2ad-ld2*(q2vm-q2vd);
 q3vr=q3vd-ld3*(q3pm-q3pd);  q3ar=q3ad-ld3*(q3vm-q3vd);
 q4vr=q4vd-ld4*(q4pm-q4pd);  q4ar=q4ad-ld4*(q4vm-q4vd);
 q5vr=q5vd-ld5*(q5pm-q5pd);  q5ar=q5ad-ld5*(q5vm-q5vd);

 ss1=q1vm-q1vr;  ss2=q2vm-q2vr;  ss3=q3vm-q3vr;  ss4=q4vm-q4vr;  ss5=q5vm-q5vr;

 c1 = cos(q1pm);  c2 = cos(q2pm);  c4 = cos(q4pm);  s1 = sin(q1pm);  s2 = sin(q2pm);  s4 = sin(q4pm);
 c22  = c2*c2;  s22  = s2*s2;  c2s2  = c2*s2;  c24  = c2*c4-s2*s4;  s24  = s2*c4+c2*s4;
 c242  = c24*c24;  s242  = s24*s24;  c24s24 = c24*s24;  d3  = q3pm;  d32 = d3*d3;
 // calcul des accelerations et vitesses de references (Slotine-Li)
 q1vq1v = q1vm*q1vr;
 q1vq2v = q1vm*q2vr+q2vm*q1vr;
 q1vq3v = q1vm*q3vr+q3vm*q1vr;
 q1vq4v = q1vm*q4vr+q4vm*q1vr;
 q1vq5v = q1vm*q5vr+q5vm*q1vr;
 q2vq2v = q2vm*q2vr;
 q2vq3v = q2vm*q3vr+q3vm*q2vr;
 q2vq4v = q2vm*q4vr+q4vm*q2vr;
 q2vq5v = q2vm*q5vr+q5vm*q2vr;
 q4vq4v = q4vm*q4vr;
 q4vq5v = q4vm*q5vr+q5vm*q4vr;
```

```
// calcul des elements de la matrice de regression
  w11 = q1ar;
  w12 = c22*q1ar-c2s2*q1vq2v;
  w13 = s22*q1ar+c2s2*q1vq2v;
  w14 = s22*d32*q1ar+c2s2*d32*q1vq2v+s22*d3*q1vq3v;
  w15 = -(s22*d3*q1ar+c2s2*d3*q1vq2v+0.5*s22*q1vq3v);
  w16 = 2*s2*d3*s24*q1ar+(s2*c24+c2*s24)*d3*q1vq2v+s2*s24*q1vq3v+s2*c24*d3*q1vq4v;
  w17 = c242*q1ar-c24s24*q1vq2v-c24s24*q1vq4v;
  w18 = s242*q1ar+c24s24*q1vq2v+c24s24*q1vq4v;
  w111 = c24*q5ar-0.5*s24*(q2vq5v+q4vq5v);
  w112 = 0;   w113 = 0;   w114 = 0;    w115 = q1vr;
   w22 = c2s2*q1vq1v;     w23 = -c2s2*q1vq1v;
  w24 = d32*q2ar-c2s2*d32*q1vq1v+d3*q2vq3v-GR*s2*d3;
  w25 = -(d3*q2ar-c2s2*d3*q1vq1v+0.5*q2vq3v-0.5*GR*s2);
  w26=2*c4*d3*q2ar-s4*q3ar+c4*d3*q4ar-(s2*c24+c2*s24)*d3*q1vq1v+c4*q2vq3v-
        s4*d3*(q2vq4v+q4vq4v)*GR*s24;
  w27 = c24s24*q1vq1v;        w28 = -c24s24*q1vq1v;     w29 = q2ar;    w210 = q4ar;
  w211 = 0.5*s24*q1vq5v;      w212 = -GR*s2;    w213 = 0;    w214 = 0;    w216 = q2vr;

  w34 = q3ar-s22*d3*q1vq1v-d3*q2vq2v+GR*c2;     w35 = 0.5*s22*q1vq1v+0.5*q2vq2v;
  w36 = -s4*(q2ar+q4ar)-s2*s24*q1vq1v-c4*(q2vq2v+q2vq4v+q4vq4v);
  w314 = -0;    w317 = q3vr;
  w46 = c4*d3*q2ar-s4*q3ar-s2*c24*d3*q1vq1v+s4*d3*q2vq2v+c4*q2vq3v-GR*s24;
  w47 = c24s24*q1vq1v;      w48 = -c24s24*q1vq1v;    w410 = q2ar+q4ar;
  w411 = 0.5*s24*q1vq5v;   w413 = 0;    w418 = q4vr;
  w511 = c24*q1ar+q5ar-0.5*s24*(q1vq2v+q1vq4v);    w519 = q5vr;
   // /mis a jour des parameters estimes
  p1ed  = migm[0] * ss1*w11;
  p2ed  = migm[1] *(ss1*w12 +ss2*w22);
  p3ed  = migm[2] *(ss1*w13 +ss2*w23);
  p4ed  = migm[3] *(ss1*w14 +ss2*w24 +ss3*w34);
  p5ed  = migm[4] *(ss1*w15 +ss2*w25 +ss3*w35);
  p6ed  = migm[5] *(ss1*w16 +ss2*w26 +ss3*w36 +ss4*w46);
  p7ed  = migm[6] *(ss1*w17 +ss2*w27        +ss4*w47);
  p8ed  = migm[7] *(ss1*w18 +ss2*w28        +ss4*w48);
  p9ed  = migm[8] *         ss2*w29;
  p10ed = migm[9] *(        ss2*w210        +ss4*w410);
  p11ed = migm[10]*(ss1*w111+ss2*w211       +ss4*w411+ss5*w511);
  p12ed = migm[11]*(ss1*w112+ss2*w212);
```

```
p13ed = migm[12]*(ss1*w113+ss2*w213        +ss4*w413);
p14ed = migm[13]*(ss1*w114+ss2*w214+ss3*w314);
p15ed = migm[14]* ss1*w115;
p16ed = migm[15]*        ss2*w216;
p17ed = migm[16]*              ss3*w317;
p18ed = migm[17]*                    ss4*w418;
p19ed = migm[18]*                          ss5*w519;
zf[0]=p1ed;  zf[1]=p2ed;  zf[2]=p3ed;  zf[3]=p4ed;  zf[4]=p5ed;  zf[5]=p6ed;  zf[6]=p7ed;  zf[7]=p8ed;
zf[8]=p9ed;  zf[9]=p10ed;  zf[10]=p11ed;  zf[11]=p12ed;  zf[12]=p13ed;  zf[13]=p14ed;  zf[14]=p15ed;
zf[15]=p16ed;  zf[16]=p17ed;  zf[17]=p18ed;  zf[18]=p19ed;
// calcul des éléments de la matrice B
 aak = c2s2*(-p2e+p3e+d32*p4e-d3*p5e)+p6e*d3*(s2*c24+c2*s24)+c24s24*(p8e-p7e);
bbk = s22*(p4e*d3-0.5*p5e)+p6e*s2*s24;    cck = p6e*s2*c24*d3+c24s24*(p8e-p7e);
ddk = p11e*s24;    eek = p4e*d3-0.5*p5e+p6e*c4;    ffk = p6e*s4*d3;    ggk = p6e*c4;

// calcul des éléments de la matrice d'inertie H
He11 = p1e+p2e*c22+p3e*s22+p4e*s22*d32-p5e*s22*d3+2*p6e*s2*d3*s24+p7e*c242+p8e*s242;
He15 = p11e*c24;    He22 = p9e+(p4e*d32)-(p5e*d3)+2*p6e*d3*c4;    He23 = -p6e*s4;
He24 = p10e+p6e*d3*c4;    He33 = p4e;    He34 = -p6e*s4;    He44 = p10e;    He55 = p11e;
He1 = He11*q1ar+He15*q5ar;    He2 = He22*q2ar+He23*q3ar+He24*q4ar;
He3 = He23*q2ar+He33*q3ar+He34*q4ar;    He4 = He24*q2ar+He34*q3ar+He44*q4ar;
He5 = He15*q1ar+He55*q5ar;
// calcul des éléments de la matrice de coriolis V
Ve1 = aak*q1vq2v+bbk*q1vq3v+cck*q1vq4v-0.5*ddk*(q2vq5v+q4vq5v);
Ve2 = 0.5*ddk*q1vq5v+eek*q2vq3v-ffk*(q2vq4v+q4vq4v)-aak*q1vq1v;
Ve3 = -ggk*(q2vq4v+q4vq4v)-bbk*q1vq1v-eek*q2vq2v;
Ve4 = 0.5*ddk*q1vq5v+ggk*q2vq3v-cck*q1vq1v+ffk*q2vq2v;
Ve5 = -0.5*(q1vq2v+q1vq4v)*ddk;
// calcul des éléments de la matrice de gravite G
Ge1 = 0;    Ge2 = -GR*(p12e*s2-0.5*p5e*s2+p4e*s2*d3+p6e*s24);    Ge3 = GR*p4e*c2;
Ge4 = -GR*p6e*s24;
 // calcul des éléments de la matrice de frottements
Fe1 = p15e*q1vr;    Fe2 = p16e*q2vr;    Fe3 = p17e*q3vr;    Fe4 = p18e*q4vr;    Fe5 = p19e*q5vr;
// calcul des couples selon le modèle dynamique
couple[0] = He1 + Ve1 + Ge1 + Fe1 - Kd1*ss1;
couple[1] = He2 + Ve2 + Ge2 + Fe2 - Kd2*ss2;
couple[2] = He3 + Ve3 + Ge3 + Fe3 - Kd3*ss3;
couple[3] = He4 + Ve4 + Ge4 + Fe4 - Kd4*ss4;
couple[4] = He5 + Ve5 + 0.0 + Fe5 - Kd5*ss5;
```

```
p1e = pr[0];  p2e = pr[1];  p3e = pr[2];  p4e = pr[3];  p5e = pr[4];  p6e = pr[5];  p7e = pr[6];
p8e = pr[7];  p9e = pr[8];  p10e = pr[9];  p11e = pr[10];  p12e = pr[11];  p13e = pr[12];  p14e = pr[13];
p15e = pr[14];  p16e = pr[15];  p17e = pr[16];  p18e = pr[17];  p19e = pr[18];
aak = c2s2*(-p2e+p3e+d32*p4e-d3*p5e)+p6e*d3*(s2*c24+c2*s24)+c24s24*(p8e-p7e);
bbk = s22*(p4e*d3-0.5*p5e)+p6e*s2*s24;
cck = p6e*s2*c24*d3+c24s24*(p8e-p7e);
ddk = p11e*s24;
eek = p4e*d3-0.5*p5e+p6e*c4;
ffk = p6e*s4*d3;
ggk = p6e*c4;

He11 = p1e+p2e*c22+p3e*s22+p4e*s22*d32-p5e*s22*d3+2*p6e*s2*d3*s24+p7e*c242+p8e*s242;
He15 = p11e*c24;  He22 = p9e+(p4e*d32) -(p5e*d3)+2*p6e*d3*c4;  He23 = -p6e*s4;
He24 = p10e+p6e*d3*c4;  He33 = p4e;  He34 = -p6e*s4;  He44 = p10e;  He55 = p11e;

q1vq1s = q1vm*q1vm;
q1vq2s = q1vm*q2vm+q2vm*q1vm;
q1vq3s = q1vm*q3vm+q3vm*q1vm;
q1vq4s = q1vm*q4vm+q4vm*q1vm;
q1vq5s = q1vm*q5vm+q5vm*q1vm;

q2vq2s = q2vm*q2vm;
q2vq3s = q2vm*q3vm+q3vm*q2vm;
q2vq4s = q2vm*q4vm+q4vm*q2vm;
q2vq5s = q2vm*q5vm+q5vm*q2vm;

q4vq4s = q4vm*q4vm;
q4vq5s = q4vm*q5vm+q5vm*q4vm;

Vse1 = aak*q1vq2s+bbk*q1vq3s+cck*q1vq4s-0.5*ddk*(q2vq5s+q4vq5s);
Vse2 = 0.5*ddk*q1vq5s+eek*q2vq3s-ffk*(q2vq4s+q4vq4s)-aak*q1vq1s;
Vse3 = -ggk*(q2vq4s+q4vq4s)-bbk*q1vq1s-eek*q2vq2s;
Vse4 = 0.5*ddk*q1vq5s+ggk*q2vq3s-cck*q1vq1s+ffk*q2vq2s;
Vse5 = -0.5*(q1vq2s+q1vq4s)*ddk;

Ge1 = 0;  Ge2 = -GR*(p12e*s2-0.5*p5e*s2+p4e*s2*d3+p6e*s24);  Ge3 = GR*p4e*c2;  Ge4 = -GR*p6e*s24;

Fe1 = p15e*q1vm;  Fe2 = p16e*q2vm;  Fe3 = p17e*q3vm;  Fe4 = p18e*q4vm;  Fe5 = p19e*q5vm;
```

```
  cple[0] = Vse1 + Ge1 + Fe1;
  cple[1] = Vse2 + Ge2 + Fe2;
  cple[2] =  Vse3 + Ge3 + Fe3 ;
  cple[3] =  Vse4 + Ge4 + Fe4;
  cple[4] =  Vse5 + 0.0 + Fe5;

  mi11=He55/(He11*He55-He15*He15);
  mi15= -He15/(He11*He55-He15*He15);
   mi22=(He23*He23-He33*He44)/(He22*He23*He23-He22*He33*He44-
          2*He23*He23*He24+He33*He24*He24+He23*He23*He44);
mi23=-(-He44+He24)*He23/(He22*He23*He23-He22*He33*He44-
          2*He23*He23*He24+He33*He24*He24+He23*He23*He44);
mi24=-(He23*He23-He24*He33)/(He22*He23*He23-He22*He33*He44-
        2*He23*He23*He24+He33*He24*He24+He23*He23*He44);
mi33=-(He22*He44-He24*He24)/(He22*He23*He23-He22*He33*He44-
        2*He23*He23*He24+He33*He24*He24+He23*He23*He44);
mi34=(He22-He24)*He23/(He22*He23*He23-He22*He33*He44-
          2*He23*He23*He24+He33*He24*He24+He23*He23*He44);
mi44=-(He22*He33-He23*He23)/(He22*He23*He23-He22*He33*He44-
       2*He23*He23*He24+He33*He24*He24+He23*He23*He44);
mi55=He11/(He11*He55-He15*He15);

  vec1 = couple[0]-cple[0];
  vec2 = couple[1]-cple[1];
  vec3 = couple[2]-cple[2];
  vec4 = couple[3]-cple[3];
  vec5 = couple[4]-cple[4];

dvm1= mi11*vec1+mi15*vec5 - q1ad;
dvm5= mi15*vec1+mi55*vec5 - q5ad;
dvm2 = mi22*vec2 +mi23*vec3 +mi24*vec4 -q2ad;
dvm3 = mi23*vec2 +mi33*vec3 +mi34*vec4 - q3ad;
dvm4 = mi24*vec2 +mi34*vec3 +mi44*vec4 -  q4ad;
// ecriture des sorties
   zf[19]=yf[24];   zf[20]=yf[25];  zf[21]=yf[26];  zf[22]=yf[27];  zf[23]=yf[28];
  zf[24]=dvm1; zf[25]=dvm2;  zf[26]=dvm3;  zf[27]=dvm4;  zf[28]=dvm5;
ep[19]=q1pd; ep[20]=q2pd; ep[21]=q3pd; ep[22]=q4pd; ep[23]=q5pd;
ep[24]=q1vd; ep[25]=q2vd; ep[26]=q3vd; ep[27]=q4vd; ep[28]=q5vd;
```

```
for(i=0;i<NB_PAR; i++) ep[i]=pr[i];
```

Programme d'implantation en temps réel

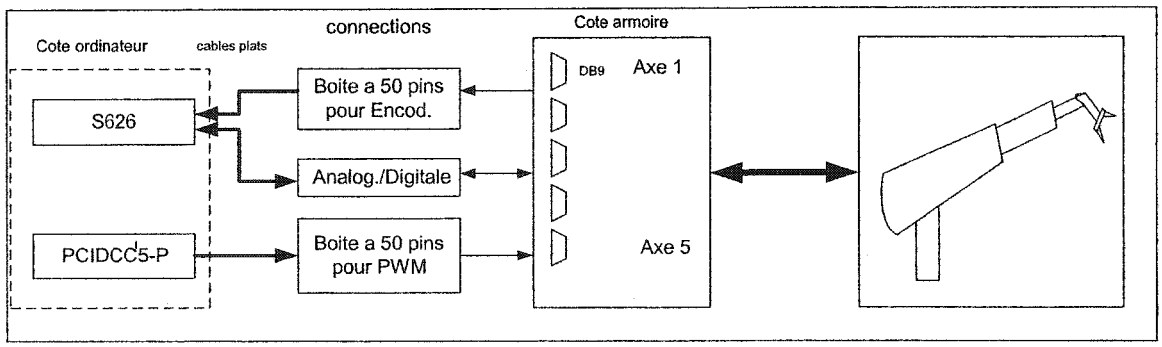
```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//        PROGRAMME D IMPLENTATION EN TEMPS REEL DE LA COMMANDE PID
//        APPLIQUEE  AU ROBOT US MAKER 100 A 5 AXES
//        CE PROGRAMME  ENVOIT LES DONNEES DU RESULTAT A L INTERFACE GRAPHIQUE
//        FAIT PAR ABDELKADER HASSANE POUR LA MAITRISE EN GPA A L ETS  (2001-2002)
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

On commence par une lecture des paramètres de contrôle, ensuite une initialisation des deux cartes I/O, et on envoie les signaux de commandes sous forme de PWM :

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                BOUCLE DE COMMANDE DE PREPARATION DE L EFFECTEUR
//                DE SON POINT DE REPOS AU PREMIER SOMMET DU TRIANGLE
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

 for (tps=0;tps< 4.00;tps=tps+Ts)
  {

     teta1=q_depart[0];  vitd1=0; // consigne de position  et vitesse
      erp=teta1-qpm[0];  // erreur de position
      erv=(vitd1-vit1); // erreur de vitesse
      Kd=.2;Kp=136.00;
      vd1=(Kd*erv+Kp*erp);
      dc1=vd1/27.5;

      /////// loi de la saturation de PWM
      if ( dc1 > .89 ) { direction[0]=1;  rc1=.96; }
      if ( dc1 < -.89 ) { direction[0]=0;  rc1=.96; }
       if ( dc1 < .89  && dc1 >= 0) { direction[0]=1;  rc1=dc1+.0310399; }
```

```
    if ( dc1 > -.89  && dc1 < 0) { direction[0]=0;  rc1=-dc1+.0310399; }
// envoi de la commande PWM, directions et break
    SetDccCardFrequencyAndDutyCycle( boardAddr, chipNbr, 3,  frequency, rc1  );
    digOutValue=direction[0]+2*direction[1]+4*direction[2]+8*direction[3]+16*direction[4]+32+64+128;
```

Montage des cartes et des connections lors de l'expérience.



Connections de DB9 côté armoire

Pin	Description
1	Intensité du courant
4	Frein du moteur
5	Masse
6	PWM
7	Encodeur canal A
8	Encodeur canal B
9	Sens de rotation

Connections de PWM (PCIDCC5-P)

Pin	Description
31	Axe 1
32	Axe 2
33	Axe 3
34	Axe 4
35	Axe 5
20	+5 Volt
11	Masse

Connections de S626-encodeures

Pin	Description
2	Canal A de l'axe 1
5	Canal B de l'axe 1
3, 9, 15, 21	Masse
6, 12, 18, 24	+5 Volt
11	Canal A de l'axe 2
14	Canal B de l'axe 2
20	Canal A de l'axe 3
23	Canal B de l'axe 3
28	Canal A de l'axe 4
31	Canal B de l'axe 4
37	Canal A de l'axe 5
40	Canal B de l'axe 5

BIBLIOGRAPHIE

1 Brady, M. (1982). *Trajectory Planning. Robot Motion : Planning and Control.* M. Brady et al. Cambridge, Mass. *:* MIT Press.

2 Craig, J.J. (1986). Introduction to Robotics- Mecanics and Control. *Allison-Wesley* (2^e edition).

3 Chernousko, F. L., Bolotnik, N. et Gradetsky, V. (1994). *Manipulation Robots : Dynamics, control and Optimization.* Boca Raton, Flor.: CRC Press.

4 Das A. K., Fierro R., Kumar V., Southall B., Spletzer J., et Taylor C. J. (2001). *Real-Time Vision-Based Control of a nonholonomic mobile robot. IEEE, International Conferences on Robotics and Automation,* Korea .

5 Dessaint, L.-A, Saad, M., Hebert, B. et Al-Haddad, K. (1992). *An Adaptive Controller for a Direct-Drive Scara Robot. IEEE Transcations on industrial Electronics.*

6 Goldberg, D. S. (1991). *Matrix Theory with Applications.* New York, N. Y. : McGraw-Hill.

7 Golub, G. et Ortega, J. M. (1993). *Scientific Computing : An Introduction with Parallel Computing.* Sandiego (Calif.) : Academic press.

8 Koivo, A. J. (1989). *Fundamentals for Control of Robotic manipulators.* New York (N.Y.) : John Wiley & Sons.

9 Laroussi K., Hemami H., Goddard R. (1988). *Coordination of two planar robots in Lifting.* IEEE Journal of robotics and automation, vol. 4, N0. 1, Fevrier .

10 Latombe, J.-C. (1991). *Robot Motion Planning.* Boston (Mass.) : Kluwer Academic Publishers.

11 Loffler M., Costescu N., Dawson D. (1998). *A Real-Time PC Based Control Environment.* IEEE (Philadelphia).

12 Lozano R., Taoutaou D. (2001). *Commande adaptative et applications* : Hermes, (Paris).

13 MacKerrow, P. J. (1995). *Introduction to Robotics.* Reading (Mass.) : Allisson-Wesley.

14 Paul, R. P. C. (1979). *Manipulator Cartesian Path Control.* IEEE Transactions on Systems, Man, and Cybernetics, 9, 702-711.

15 Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., *Numerical Recipes in C.* New York (N. Y.) : Cambridge University Press.

16 Richard M. Murray, Zexiang Li, S. Shankar Sastry, (1994). *A mathematical introduction to robotic manipulation :* CRC press, (Florida).

17 Saad, Y. (1992). *Numerical Methods for Large Eigenvalue Problems : Theory and Algorithms:* Manchester University Press.

18 Schiling, R. J. (1990). *Fundamentals of Robotics – Analysis and Control.* Englewood Cliffs (N.J.) : Prentice-Hall.

19 Sciavicco L., Siciliano B. (2000). *Modeling and control of Robot Manipulators*: Springer (New York).

20 Slotine, J.-J. E. et Li, W. (1991). *Applied non Linear Control. Englewood Cliffs:* (N.J.) : Prentice-Hall.

21 Slotine, J.-J. E. et Li, W. (1987). *Adaptative Manipulator Control : A Case Study.* IEEE Transcations on Automatic Control, 33(11), pp. 995-1003

22 Slotine, J.-J. E. et Li, W. (1987). *Adaptative Strategies in Constrained Manipulation.* IEEE International Conference on Robotics and Automation, 595-601.

23 Subbarao K., Verma A., Junkins J. L., (2001). *Model reference adaptive control of constrained cooperative manipulators.* Conferences on control applications, Septembre(Mexique).

24 Sugar T. G., Kumar V. (2002). *Control of cooperating mobile manipulators*. IEEE transactions on robotics and automation, vol. 18, no. 1.

25 Sugar T. G., Kumar V., Pfreundschuh G. H. (1991). *Design and Control of 3 DOF in-parallel Actuated Manipulator*. IEEE, International Conferences on Robotics and Automation, (California) – April .

26 Taylor, R. H. (1979). *Planning and Execution of Stright-Line Manipulator Trajectories*. IBM Journal of Research and Development, 23, 424-436

27 Watkins, D. S. (1991). *Fundamentals of Matrix Computations*. New York (N. Y.) : John Wiley & Sons.

28 Whitney, D. E. (1972). *The Mathematics of Coordinated Control of Prosthetics Arms and Manipulators.* Journal of Dynamic Systems, Measurement, and Control, 303-309.

29 YUE S., HENRICH D. (2002). *Manipulating deformable linear objects: Sensor-Based Fast Manipulation during Vibration.* IEEE, International Conferences on Robotics and Automation,(Washington D.C.) – May.

Sites web:

30 QNX Software Ltd. *Documentation's Guide*. Site official de QNX [En ligne]. *http://www.qdn.com*. (Page consultée le 10 novembre 2001).

31 Mabilleau P. *Système en temps réel.* Notes de cours. [En ligne] *www.gel.usherb.ca/mab/GEI455/notes/definition.pdf*. (Page consultée le 7 septembre 2002).

32 Philippe M.. *Le temps réel*. [En ligne] *http://perso.wanadoo.fr/philippe.michel*. (Page consultée le 10 novembre 2001).

33 Ripoll I.. *Linux Temps Réel (RT-LINUX)*. [En ligne] http://*mercury.chem..pitt.edu/*tiho/linuxFocus/aiscFran/May1998/article44.html (Page consultée le 10 novembre 2001).

34 Gangloff J. *Asservissements visuels rapides d'un robot.* [En ligne] http://hp2gra.u-strasbg.fr/library/thesis/gangloff. . (Page consultée le 7 septembre 2002).

35 Salvetti D. *Construction d'un système d'exploitation GNU/RT Linux.* [En ligne] http://www.crans.ens-cachan.fr/*salvetti/systeme/systeme004.html.. (Page consultée le 7 septembre 2002).

36 Blachier W. *Le temps réel* . Institut National Polytechnique de Grenoble [En ligne] http://www-ensimag.imag.fr/. (Page consultée le 10 octobre 2001).

37 Skivée F. *Systèmes embarqués* : Université de Liège Institut Montefiore Belgique [En ligne] www.montefiore.ulg.ac.be/ (Page consultée le 20/ novembre 2002).

38 Abrial B. *Définitions du temps réel* Web et administration [En ligne] www.tvtsii.net/sections.php3. (Page consultée le 29 Décembre 2002).